HAAGA-HELIA
ammattikorkeakoulu

Yritys X:n sovellustestausprosessin kehittäminen

Jussi Sandholm

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2015

HAAGA-HELIA ammattikorkeakoulu

Tiivistelmä

11.5.2015

Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Jussi Sandholm	**Ryhmätunnus tai aloitusvuosi** 2011
Raportin nimi Yritys X:n sovellustestausprosessin kehittäminen	**Sivu- ja liitesivumäärä** 44 + 3
Opettajat tai ohjaajat Petteri Puurunen	

Tämän opinnäytetyön tarkoituksena oli tutkia sovellustestausprosessia ja selvittää sen nykyisiä puutteita. Aiheena oli sovellustestaus ja sen hallinta kohdeyrityksen näkökulmasta. Työ rajattiin vain kohdeyrityksen sovellustestaukseen. Toimeksiantaja haluaa pysyä anonyymina, joten kohdeyritystä kutsutaan nimellä yritys X.

Opinnäytetyön tavoitteena oli tehdä kohdeyrityksen ohjelmistotestausprosessista määrämuotoisempaa ja korkealaatuista selkeyttämällä testausstrategia ja testauskäytännöt koko tietohallinto-organisaatiolle.

Tutkimus tehtiin haastattelemalla kohdeyrityksessä sovellustestaukseen osallistuneita henkilöitä. Tutkimusmenetelmänä käytettiin teemahaastattelua. Opinnäytetyö tehtiin keväällä 2015. Tutkimuksen tuloksena toimeksiantajalle luotiin yleinen sovellustestauksen prosessikuvaus, ohjeet testausdokumenttien täyttämiseksi ja pohjat testauspolitiikalle ja –strategialle.

Parhaiten ohjelmistojen laatu varmistetaan, kun testitapausten suunnittelu aloitetaan yhdessä sovellustyön tilaajan kanssa jo määrittelyvaiheessa. Testaukseen liittyvillä dokumenteilla tulisi olla yksilöllinen tunniste, sijaita samassa paikassa ja samanlaisessa muodossa mikäli mahdollista.

Asiasanat
Strategia, testaus, ohjelmistokehitys, suunnittelu

HAAGA-HELIA
University of Applied Sciences

Abstract

11.5.2015

Degree programme in information Technology

Authors Jussi Sandholm	**Group or year of entry** 2011
The title of thesis The software testing process development of company X	**Number of pages and appendices** 44 + 3
Supervisor(s) Petteri Puurunen	

The purpose of this thesis was to study the company's software testing process and solve its current defects. The subject of this thesis is software testing and test management from the target company's point of view. This thesis was outlined to cover only the testing within the target company. The client company wants to be anonymous and due to that it is called company X.

The aim of this thesis was to make the target company's software testing process more specified and ensure high quality by clarifying the testing strategy and testing practices to the whole information management organization.

This study was conducted by interviewing testing personnel in the target company. The method used was theme interview. The thesis was written during spring 2015. A new general testing process and a detailed guide for testing documents were created as a result of the thesis. Also the basic forms for testing policy and testing strategy were created.

The best way to ensure software quality is to start test case planning together with the customer already in the definition stage. All the testing related documentation should have a unique identification, be located in the same place and preferably be on a specific format.

Key words
Strategy, testing, software testing, planning

Sisällys

1 Johdanto

Yritykset menettävät vuosittain useita kymmeniä miljardeja puutteellisen ohjelmistotestauksen aiheuttamina suorina tai välillisinä tappioina. Ohjelmistojen testaukseen liittyvien toimenpiteiden arvioidaan vievän noin 50 prosenttia ohjelmistoprojektien kokonaisbudjetista. Mahdollisen virheen löytäminen ja korjaaminen ohjelmaa suunniteltaessa maksaa vain murto-osan siitä, mitä se maksaisi julkaisun jälkeen tehtynä. Näiden edellä mainittujen esimerkkien on tarkoitus korostaa ohjelmistotestauksen tärkeyttä. Hyvin suunniteltuna ja toteutettuna ohjelmistotestauksella varmistetaan, että lopputuloksena syntyvät ohjelmistot ovat riittävän laadukkaita ja toimivat kuten on haluttu. Suunnitelmallisella ja tehokkaalla ohjelmistotestauksella voidaan myös säästää merkittävästi ohjelmistotuotannon kokonaiskustannuksissa. Testaustyö on myös kannattavuuden kannalta mitattuna tärkein työvaihe. Tuotteiden kannattavuutta ja testausta keskenään vertailtaessa näillä on selkeä yhteys. Huolellisesti tuotteensa testaavat yritykset saavat tuotteillensa paremman katteen kuin huonosti testauksen hoitavat yritykset. Lisäksi asiakaskato ja maine huonoja tuotteita valmistavana yrityksenä korostavat tätä eroa vieläkin enemmän. (Kasurinen 2013, 11-12; Kit 1995; Tassey 2002.)

Ohjelmistotuotannon tavoitteena on valmistaa ohjelmisto, joka toimii määriteltyjen vaatimusten mukaisesti. Määrittelyt eli ohjelmalta vaadittavat toiminnot laatii yleensä taho, jonka käyttöön ohjelmisto tulee. Ohjelmistotestauksella taas tarkoitetaan, että ohjelmisto toimii määritysten ja vaatimusten mukaisesti. Koska kokonaisvaltainen testaus on ohjelmistojen monimutkaisuuden vuoksi lähes kaikissa tapauksissa mahdotonta, on testaukselle määriteltävä prioriteetit. Järjestelmän kriittiset toiminnot on testattava erityisen tarkasti. Päätös testattavista toiminnoista ja lähestymistavoista tulee olla tarkoin harkittua, jotta voidaan saavuttaa mahdollisimman hyvä optimointi ohjelman toimivuuden ja testien kattavuuden suhteen. Myös aika on ohjelmistoprojekteissa usein rajallinen, eikä kaikkia mahdollisia tapauksia ei voida testata. Tämä on otettava huomioon myös testauksen suunnittelussa. Jotta testaaminen olisi järkevää ja taloudellista, tarvitaan sekä suunnitelmallisuutta että testausstrategia. (Glenford, Sandler & Badgett 2011, 8.)

Seuraavaksi muutama esimerkki, mitä kaikkea voi tapahtua, kun testaus tehdään puutteellisesti ja ohjelmistoihin jää kriittisiä virheitä. Persianlahden sodan aikaan helmikuussa 1991 yhdysvaltalaisten ampuma ohjus harhautui maalistaan ja osui omaan parakkiin surmaten 28 ja haavoittaen yli sataa sotilasta. Tämä aiheutui ohjelmointivirheen aiheuttamasta viasta ohjuksen ohjausjärjestelmän ajastuksessa. (Blair, Obenski & Bridickas 1992.)

Kesäkuun 4. päivä vuonna 1996 Kouroussa, Ranskan Guayanassa sijaitsevasta avaruuskeskuksesta laukaistiin ARIANE 5- niminen kantoraketti, jonka oli tarkoitus kuljettaa tietoliikennesatelliitteja maata kiertävälle kiertoradalle. Raketti jouduttiin räjäyttämään noin 40 sekuntia laukaisun jälkeen, sillä ohjelmistovirheen seurauksena raketti ei enää totellut ohjausta. Räjähdyksessä menetetyn laitteiston arvoksi ilmoitettiin 500 miljoonaa dollaria. Kun tähän lisätään kymmenen vuoden ja 7 miljardin dollarin arvoinen kehitystyö, niin puhutaan varmasti yhdestä rahallisesti kalleimmasta ohjelmistovirheestä kautta aikain. (Lions 1996.)

Lääketieteen käytössä olevat järjestelmät tulee testata ja dokumentoida erityisen tarkasti. Tämäkin on opittu vasta muutaman vakavan virheen seurauksena. Therac-25 -niminen sädehoitolaite antoi hoitojen yhteydessä kuudelle ihmiselle tappavan annoksen säteilyä vuosina 1985–87. Potilaat saivat jopa 100 kertaa tarkoitettua enemmän säteilyä, koska virransyöttöä ohjasi viallinen ohjelmisto. Toinen vastaava tapaus sattui Panama Cityssä vuonna 2000, jossa Yhdysvaltalaisen Multidatan hoitosuunnitteluohjelma antoi potilaille säteilyä 20–100 prosenttia yli tarvittavan annoksen. Tämä virhe ilmeni, kun ohjelman tarvitsevat tiedot syötettiin eri järjestyksessä. Liian suuret säteilyannostukset tappoivat suoraan ainakin 5 ihmistä. (International Atomic Energy Agency 2001; Leveson & Turner 1993.)

Voimme vain kuvitella, mitä seuraavien esimerkkien seurauksena olisi voinut tapahtua. Tapahtumat ovat kylmän sodan ajalta, jolloin Yhdysvallat ja silloinen Neuvostoliitto olivat jatkuvassa valmiudessa ydinaseidensa kanssa. Vuonna 1980 NORAD (North American Air Defense Command) ilmoitti virheellisesti, että Yhdysvallat on ohjushyökkäyksen kohteena. Tämän virheellisen ilmoituksen aiheutti virtapiirissä ollut vika, jota ohjelmisto ei ollut ottanut huomioon. Virhe onneksi havaittiin ennen

vastahyökkäyksen täytäntöönpanoa. Ohjelmistovirheet olisivat siis voineet aiheuttaa kolmannen maailmansodan jo kymmeniä vuosia sitten. Toinen vastaavanlainen tapaus sattui 1983 neuvostoliittolaisille, kun heidän satelliittinsa varoitti tulevista yhdysvaltalaisohjuksista. Päivystävä upseeri ei kuitenkaan tätä hälytystä uskonut. Hän luotti omaan vaistoonsa ja siihen, että kyseessä on väärä hälytys. Jos toinen osapuoli olisi reagoinut vastahyökkäyksellä ja käyttänyt ydinasetta, eläisimme varmasti hyvin erilaisessa maailmassa nykyään. (U.S General Accounting Office 1981; Washingtonpost 1999.)

1.1 Työn tausta

Opinnäytetyö tehdään toimeksiantona yritys X:lle. Työn tarkoituksena on kartoittaa ja analysoida sovelluskehityksen nykytilaa sekä kehittää yritys X:n tietohallinnon nykyisiä testauskäytäntöjä ja -menetelmiä. Nykytilaselvityksen jälkeen luodaan hyvien käytäntöjen perusteella yleisesti sovellustestaamiseen sopiva testauspolitiikka ja siitä johdettu yleinen testausstrategia. Yritys X:n tietohallinnossa ei ole omaa ohjelmistotuotantoa, vaan varsinaisen ohjelmistokehityksen ja alkupään testauksen tekee järjestelmätoimittaja. Tämä opinnäytetyö rajataan koskemaan vain yritys X:n tietohallinnossa tehtävää sovellustestausta, siihen liittyviä valmisteluja ja sen hallintaa. Teoriaosuudessa käsitellään ohjelmistotestausta yleisesti, jotta kokonaiskuvan muodostaminen aiheesta olisi helpompaa. Monesti yritykset valitsevat heille huonosti sopivan mallin. Sopimattoman mallin tarkka noudattaminen käy jossain vaiheessa haastavaksi ja jossain vaiheesta tästä luovutaan. Organisaatioiden tulisikin valita heille soveltuva testausmalli, jota noudattaa. (Kasurinen 2013, 42.)

Yritys X on vajaan tuhannen työntekijän yritys, jonka tietohallinnossa työskentelee noin 80 henkilöä.

1.2 Tutkimusongelma ja -menetelmät

Tämän opinnäytetyön tarkoituksena on yhtenäistää olemassa olevat yritys X:n tietohallinnon testauskäytännöt, paikallistaa kehitystä tarvitsevat kohdat, kirjata testauspolitiikka ja luoda yleinen testausstrategia, jota voidaan yleisesti soveltaa kaikkeen yritys X:n tekemään sovellustestaukseen. Uusi testauspolitiikka sekä yrityksen käytössä olevat

hyvät käytännöt dokumentoidaan jatkokehitystä varten. Luottamuksellisista syistä vain osa tuloksista voidaan esitellä tuloksissa ja liitteissä. Työssä huomioidaan yritys X:lle tärkeän SAP-ohjelmiston testaamiseen liittyvät erityishuomiot ja käytännöt. Myös testaukseen liittyvä materiaali on mukana muutosprosessissa. Työssä keskitytään yritys X:n tietohallinnossa tehtävään sovellustestaamiseen ja sen hallintaan, eikä sen järjestelmätoimittajien harjoittamaan testaamiseen oteta kantaa.

Tavoitteena on kehittää malli ja käytännöt, jotka tekevät ohjelmistotestauksesta määrämuotoisempaa, selkeyttävät testausstrategian ja varmistavat yritys X:n käyttöön tulevien ohjelmistotuotteiden luotettavuuden, käytettävyyden ja korkean laadun. Työn tuloksena tietohallinnon testaustavat yhtenäistyvät ja testauspolitiikka, kuten strategiakin ovat kirjallisena kaikkien niitä tarvitsevien saatavilla. Työ kuvaa nykytilan haastattelujen ja nykyisten testausdokumenttien perusteella. Nykytilasta johdetun ja yhtenäistetyn yleisen testauspolitiikan ja testausstrategian soveltuvuutta tarkastellaan erityisesti SAP-testauksen näkökulmasta, joka on yritys X:lle eniten testaustyötä teettävä järjestelmä. Työtä varten ohjelmistotestauksen teoriaa ja käytäntöjä on kerätty internet-tietolähteistä ja ohjelmistokehityksen kirjallisuudesta. Teoriapohjan työhön tarjoavat ohjelmistokehityksessä yleisesti käytössä olevat standardit ja alan vakiintuneet hyvät käytännöt. Empiirisessä osuudessa haastatellaan tietohallinnossa työskenteleviä järjestelmäpäälliköitä, tietohallintopäälliköitä sekä yhtä järjestelmäasiantuntijaa. Työn painopiste on testauksen suunnittelussa ja hallinnassa.

Opinnäytetyö vastaa seuraaviin kysymyksiin:

- Mikä on yritys X:n sovellustestauksen nykytila?
- Kuinka yritys X:n nykyinen testausprosessi toimii?
- Tunnistetaanko nykyinen testauspolitiikka tai -strategia tietohallinnossa?
- Onko testaustyötä tekevillä käytössään tarvittavat työkalut?
- Miltä uusi testausstrategia voisi näyttää?
- Miltä uusi käytännön testausta tukeva prosessi- ja aktiviteettiluettelo voisi näyttää?

Ongelmia: Nykyisellään testauspolitiikkaa tai yleistä testausstrategiaa ei ole olemassa kirjallisena. Tämän seurauksena yhteinen näkemys puuttuu ja jokainen järjestelmän kehityksestä vastaava taho testaa ja dokumentoi omalla tavallaan. Tällaisen hajanaisen sovellustestaamisen kehittäminen on haastavaa: ”miten kehittää jotain, mitä ei ole olemassa?”. Kun käytännöt saadaan yhtenäisiksi ja kirjallisiksi, toimintaa voidaan tarkemmin mitata ja kehityskohteet ovat helpommin tunnistettavissa. Nykytila-analyysin jälkeen tunnistetaan myös, kuinka kaukana ollaan tietohallinnon tahtotilasta. Ja tiedostetaan myös se, mitä kaikkea pitää tehdä toisin tahtotilan saavuttamiseksi.

Työn aluksi perehdytään ohjelmistotestaamiseen yleisesti ja sen eri vaiheisiin. Seuraavaksi tutustutaan testauspolitiikkaan, -strategiaan ja -prosessiin sekä testauksen suunnitteluun ja hallintaan. Empiirisessä osassa haastatellaan tietohallinnon asiantuntijoita ja muita testauksesta vastaavia henkilöitä. Haastatteluilla selvitetään sovellustestauksen nykytila, etsitään ongelmakohtia ja vastataan muihin tutkimusongelmiin. Haastatteluissa esille tulleet hyvät käytännöt, ongelmakohdat sekä muut asiat kootaan yhteen ja kirjataan testauspolitiikkaan ja tästä jalostuvaan yleiseen testausstrategiaan soveltuvin osin. Viimeinen luku on omaa pohdintaani sekä jatkotutkimuskohteiden esittelyä varten. Näiden lisäksi käytetään hyväksi alan kirjallisuudesta ja muista luotettavista lähteistä saatavia käytäntöjä. Yritys X:lle luodaan yleisesti tunnistettujen hyvien käytäntöjen perusteella sopiva testauspolitiikka ja yleinen testausstrategia.

Työn tarkoituksena on siis löytää paremmat mallit sovelluskehitykseen sekä erityisesti SAPin testaamiseen. Työn loppuvaiheessa luodaan uudet testidokumenttipohjat ja muut relevantit lomakkeet. Lopputuloksena syntyvän testauspolitiikan ja yleisen testausstrategian ansiosta yritys X:n tietohallinto yhtenäistää sovellustestauksen prosessiaan. Toimintaan vaikuttavat muutokset aiheuttavat tulevaisuudessa vähemmän väärinkäsityksiä, testausprosessi on yhtenäisempi ja prosessin kokonaisvaltainen kehittäminen mahdollista. Testaamisesta saadaan selkeämpää, kun tiedetään kuka vastaa mistäkin osa-alueesta ja keneltä saa tietoa mahdollisissa ongelmatilanteissa. Kaikille on jatkossa selvää, miten meillä testataan, mitä standardeja noudatamme ja miten mittaamme laatua. Lisäksi jatkossa testausprosessin kehittymisen oletetaan näkyvän nopeampana testauksen läpimenona ja parempana laatuna resursseja säästäen.

Opinnäytetyöhön tutustuminen antaa lukijalle yleiskäsityksen testaustyöstä, sen vaatimasta huomiosta ja siitä, kuinka testausta tulisi suunnitella ja toteuttaa. Ohjelmistojen testaus on varsin epäkiitollista työtä, koska hyvin toteutetun ja onnistuneen testauksen jälkiä ei loppukäyttäjä havaitse. Sen sijaan puutteellisen testauksen jäljet näkyvät myös loppukäyttäjillä. Huonosta testauksesta kärsii lopulta koko yritys ja monesti myös yrityksen eri sidosryhmät. Työssä käytetään nimitystä asiakas, jolla viitataan yleisesti työn tilaajaan. Kehitettävät ohjelmistot tulevat pääasiassa liiketoiminnan käyttöön, mutta asiakkaalla voidaan joissakin tapauksissa tarkoittaa myös IT-organisaatiota. Liitteessä 3. on lueteltu työssä käytetyt termit, lyhenteet ja niiden määritelmät.

2 Ohjelmistotestaus ja sen hallinta

Tämä luku käsittelee yleisesti testausta ja sen hallintaa. Ohjelmiston testaaminen on osa ohjelmistotuotantoa, joka yleensä tehdään projekteina. Ohjelmistotestauksen käsikirjassa Jussi Pekka Kasurinen määrittelee testauksen lyhyesti lauseella: ”varmistetaan että tehdään oikeaa tuotetta ja että tuote on tehty oikein”. Hän lisää, ”että testaus on myös jatkuvaa vertailua suunnitellun ja tekeillä olevan työn kesken”. Tarkoituksena on tunnistaa missä mennään ja kuinka hyvin nykyinen tuotos vastaa suunniteltua. (Kasurinen 2013, 10-11.)

Nykyään tietokoneiden tehokkuus ja ohjelmistojen monipuolisuus asettaa ohjelmien testaukselle jatkuvasti kasvavia haasteita. Tilanne on hyvin erilainen kuin tietokoneiden alkuaikoina, jolloin ohjelmat olivat hyvin yksinkertaisia. Nyt testaukselta vaaditaan sekä suunnitelmallisuutta että tehokkuutta. Kaikkia mahdollisia kombinaatioita ei ohjelmien monimutkaisuuden ja laajuuden vuoksi voida testata. Tulee testata ominaisuudet, jotka ohjelmalta vaaditaan toimiakseen suunnitelman mukaisesti. Ohjelmaan saatetaan jättää tietoisestikin pieniä virheitä, jotka eivät haittaa käyttöä. Näitä virheitä voidaan korjata myöhemmin ohjelmiston ylläpitovaiheessa. Nykyisestä testaamisesta tekee haastavampaa ohjelmointikielien lukumäärä, lukuisat käyttöjärjestelmät sekä laitteiston kehittyminen. Enää on vaikea kuvitella, että kukaan selviäisi päivittäisestä työstä käyttämättä mitään jonkinlaisen ohjelmiston ohjaamaa laitetta, vaikka aivan tavallista matkapuhelinta. Monet perinteiset työkalut, kuten porakone, toimivat vielä ilman ohjelmistoa, mutta esimerkiksi monet elektroniset mittarit tarvitsevat jo ohjelmiston toimiakseen. Ohjelmistojen toimivuus on tärkeää ja toimimattomina ne saattavat maksaa ihmishenkiä, kuten edellä mainitut esimerkit osoittivat. Tietokoneet ja niitä ohjaavat ohjelmistot vaikuttavat elämäämme ja jatkuvasti enemmän myös ympärillämme olevaan yhteiskuntaan. Ohjelmistojen testaus on myös helpottunut ohjelmistokehityksen alkuajoista esimerkiksi testaustyökalujen ja -käytäntöjen kehittymisen myötä. Hyvät testauskäytännöt ja graafiset käyttöliittymät ovat tästä hyviä esimerkkejä. Ohjelmistotestaus on prosessi tai sarja prosesseja, jotka on suunniteltu varmistamaan, että koodi tekee sen, mihin se on suunniteltu, eikä tee mitään tahatonta. (Glenford, Sandler & Badgett 2011, 1-2.)

Ohjelmistot leviävät kaikkialle, esimerkiksi työkaluihin, leluihin ja kodinkoneisiin. Tulevaisuudessa tällainen ohjelmistoja käyttävien esineiden leviämistä nopeuttava asia saattaa hyvinkin olla Internet of Things eli vapaasti suomennettuna asioiden internet. Tästä käytetään myös nimitystä teollinen internet, koska teollisuus odottaa kovasti tuottavuuden parantumista sen seurauksena. Asioiden internetin kautta laitteet keskustelisivat keskenään, kuten nykyisessä internetissä ihmisetkin. Teollisuudessa voitaisiin esimerkiksi välttyä tuotantokatkoilta, kun laitteet viestisivät huoltotarpeestaan. (Perttu 2013.)

Testaaminen on tekninen tehtävä, mutta siinä vaaditaan myös taloudellisia näkökulmia sekä tietoa ihmisten käyttäytymisestä. Testaajalta vaaditaan myös oikeaa asennetta tai näkemystä onnistuneeseen testaukseen. Testaajan asenne saattaa olla jossain tapauksissa tärkeämpi kuin itse testausprosessi, sillä testaajan on haluttava löytää virheitä, jotta niitä löytyisi mahdollisimman paljon. Parhaaseen lopputulokseen päästään, kun testauksen tavoitteena on löytää ja poistaa ohjelmassa olevat virheet. Testauksen voisi siis sanoa olevan prosessi olemassa olevien virheiden löytämiseksi. Näin ohjelmistotestaus tuo ohjelmaan lisäarvoa parantamalla sen luotettavuutta, laatua sekä toimintavarmuutta. Ohjelmistotestauksen todellisen olemuksen ymmärtäminen voi olla perusteellinen tekijä mahdollisimman hyvään lopputulokseen tähdättäessä. Ihmiset ovat yleensä hyvin päämäärätietoisia, ja tavoitteiden asettamisella on tärkeä psykologinen vaikutus. Testattaessa perinteinen ajattelu on käännettävä muotoon, jossa virheen löytyminen eli toisin sanoen epäonnistunut testiajo tarkoittaakin onnistumista. Ohjelmiston testaus on lopullisesti onnistunut, kun virheitä ei enää onnistuta löytämään ja ohjelma vastaa suunniteltua. Testaaminen on epäonnistunut silloin, jos yhtään virhettä ei löydetä, sillä virheettömän ohjelmiston käsite on epärealistinen. Ohjelmien monimutkaisuuden ja yksinkertaisenkin ohjelman kaikkien tapausten lukumäärän vuoksi testaamisen on oltava taloudellista ja resursseja säästävää. (Glenford, Sandler & Badgett 2011, 5-6.)

2.1 Suunnittelu ja dokumentointi

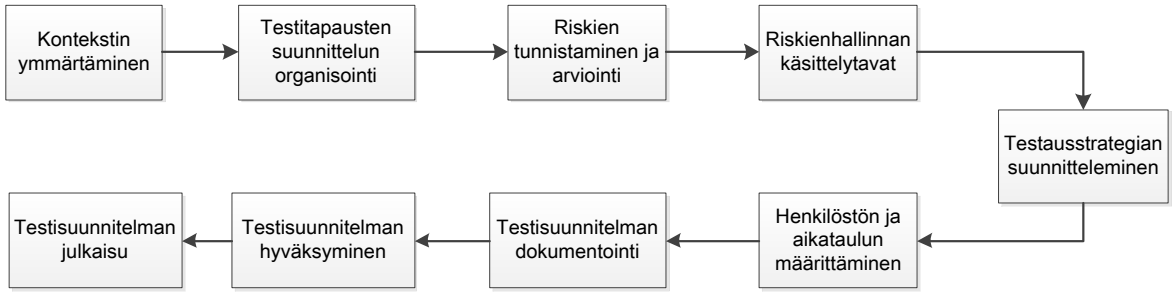
Ohjelmistotuotanto alkaa tarvittavan ohjelman määrittely- ja suunnitteluvaiheella, kuten myöhemmin v-mallin esittelyssä todetaan. Suunnitteluvaiheessa löydetyt virheet maksavat kymmenesosan kehitystyön aikana ja sadasosan julkaisun jälkeen löydettyihin verrattuna. Tutkimuksen mukaan laadun tärkein lähde on kehitys- ja suunnitteluvaihe,

ei itse testaus. Testaus on paljon muutakin kuin toiminnallisuuksien toteamista ja ohjelman käyttöä. Testauksessa kokeillaan ohjelmaa ja arvioidaan sen rakennetta, mutta siihen kuuluu myös suunnittelua, analysointia, raportointia ja tiedonhallintaa. Eri työvaiheissa keskitytään erilaisiin asioihin, ja testausmenetelmiä vaihdellen ohjelmasta pyritään löytämään virheitä erilaisin lähestymistavoin. Ohjelmistotuotannossa toteutettavasta sovelluksesta voidaan vaatimusten perusteella laatia testitapaukset sekä tehdä riskikartoitus erityisen tarkkaan testattavista komponenteista. (Kasurinen 2013, 18, 48-49.)

Psykologisten asioiden jälkeen testauksessa on tärkeää testitapausten huolellinen suunnittelu ja luominen, jotta nämä olisivat tehokkaita. Tulisikin tarkasti määritellä käyttötavat ja toiminnot, joiden läpikäyminen poistaa suurimman osan mahdollisista virheistä. Ratkaistava kysymys siis on: millä testitapausten joukolla ohjelmasta löydetään siinä olevat virheet mahdollisimman kattavasti. Kaikista huonoin testaustapa on pelkästään satunnaisten syötteiden testaaminen. Suositeltava tapa on suunnitella testitapaukset musta laatikko -testeinä, ja tehdä näille täydentävä lasilaatikkosuunnitelma, jos vain mahdollista. Käsittelen näitä testausmenetelmiä lisää myöhemmin tässä luvussa. Myös näissä testausmalleissa on puutteita, eivätkä ne ole aina mahdollisiakaan, kuten myöhemmin käy ilmi. Vanha ohjelmistotestauksen viisaus: jos luulit, että ohjelman suunnitteleminen ja koodaaminen on haastavaa, et ole nähnyt vielä mitään. (Glenford, Sandler & Badgett 2011, 41-42.)

Dokumentteja tarvitaan testauksen hallinnointiin, suunnitteluun ja testitulosten tallentamiseen. Yleensä testausta tekee useampi ihminen. Tärkeä työkalu testausta suorittavalle on valmiit dokumenttipohjat määrittelyä, raportointia ja ohjausta varten. Dokumenttien tulisi olla kaikkien saatavilla ja mielellään samassa paikassa, jotta aina tuorein versio olisi saatavilla. Testausdokumenttipohjien on tarkoitus varmistaa, että kaikki jollain tavalla testauksessa tarpeellinen tieto tulee kirjattua ylös, eikä testaajien aika mene dokumenttien luomiseen ja muotoiluun vaan he voivat tehokkaasti testata tai raportoida vikoja. Organisaatiolla olisi hyvä olla käytössä yhteinen dokumenttipohja, näin tarvittavien tietojen löytäminen dokumenteista helpompaa. (Kasurinen 2013, 88.)

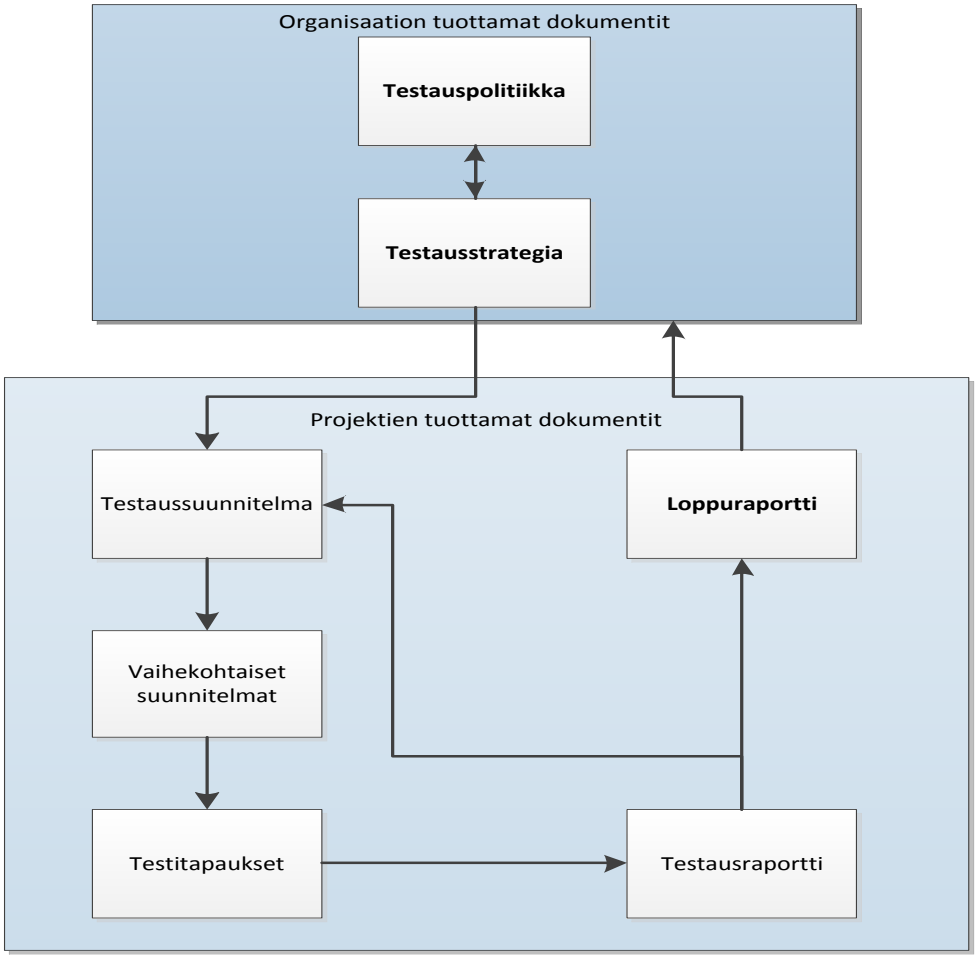
Esimerkiksi ISO /IEC 29119 standardissa on organisaatiossa oltava ainakin neljä dokumenttia: testauspolitiikka, testausstrategia, testisuunnitelma ja testausraportit. Näiden dokumenttien perusteella määritellään ja toteutetaan koko organisaation testaustoiminta, tehdään päätökset testausta koskien sekä korjataan mahdollisia epäkohtia toiminnassa. Testauspolitiikka ja – strategia ovat dokumentteja, jotka määrittelevät yleisellä tasolla kuinka testausta suoritetaan. Nämä dokumentit ovat joko ylemmän tason suunnittelemia tai sen hyväksymiä. Ylemmällä tasolla tarkoitetaan vähintään IT-johtoa. Ylätason dokumentit toimivat pelisääntöinä ja valtuuttavat testauksen projektitasolla. Testausorganisaation vastuulle jää raportointi näihin ylätason dokumenttien ongelmiin liittyen. Sekä testauspolitiikka että – strategia tulisi päivittää ja uudistaa sitä mukaa kun havaitaan tarvetta toimintatapojen kehittämisessä, muuttamisessa tai muiden ongelmien ilmaantuessa. Testauspolitiikasta ja – strategiasta kerrotaan myöhemmin tässä luvussa lisää. Testauksen suunnitteluprosessi etenee ISO/IEC/IEEE 29119-2 standardin mukaan seuraavasti:



Kuvio 1. Testauksen suunnittelun eteneminen.

Testisuunnitelma tai testaussuunnitelma on projektitason dokumentti, ja se määritellään testauspolitiikan ja – strategian pohjalta. Testisuunnitelman tarkoituksena on sovittaa politiikka ja strategiat projektin tarpeisiin huomioiden vaatimusmäärittelyt, asiakkaan toiveet ja projektin tavoitteet. Testisuunnitelma voidaan myös jakaa osiin ja puhua vaihekohtaisista suunnitelmista. Vaihekohtaisissa testaussuunnitelmissa kuvataan aina yksi testausvaihe tai testaustaso kerrallaan (moduulitestaus, integraatiotestaus jne.).
Neljäs keskeinen testauksen dokumentti on testausraportti. Testausraporteilla on tarkoituksena antaa ylimmälle johdolle tietoa, kuinka yksittäisen projektin testaustoiminta on sujunut. Näistä raporteista saattavat olla kiinnostuneita myös esimerkiksi viranomaiset ja yrityksen riskien hallinnasta vastaavat tahot. Nämä raportit toimivat myös

organisaation tiedotuskanavana, ja näiden pohjalta tehdään päätökset siitä, tuleeko testausstrategiaa muuttaa tai päivittää. Projektin päättyessä testausraporteista voidaan koostaa loppuraportti, jossa kootaan kaikki projektin tärkeimmät mittarit ja havainnot yhteen. (Kasurinen 2013, 105.)



Kuvio 2. Kuinka tavalliset testidokumentit liittyvät toisiinsa (Kasurinen 2013, 105.)

2.2 Testaaminen

Ohjelmiston testaaminen on osa ohjelmistotuotantoa. Ohjelmistoa testataan läpi koko prosessin. Tämän opinnäytetyön pääpaino on kuitenkin ohjelmistotuotannon loppupäässä, jossa ohjelmiston tai sen osan tulisi olla jo valmis, tai ainakin melkein valmis. Testaamisen tarkoituksena on varmistaa tai parantaa ohjelmiston laatu vastaamaan asiakkaan vaatimuksia. Testauksessa pyritään tapauksiin, jotka läpikäymällä on suurimmat todennäköisyydet löytää mahdollisimman monta virhettä. Tämä on ilmeinen lähesty-

mistapa testaukseen eli määrittää tapaukset, jotka mahdollistavat mahdollisimman täydellisen testaamisen. (Glenford, Sandler & Badgett 2011, 41.)

Koska kaikkien virheiden löytäminen on mahdoton, tai lähes mahdoton tehtävä, on testaus suunniteltava etukäteen, ja pohdittava käytettäviä lähestymistapoja. Kaksi vallitsevaa lähestymistapaa testaukseen ovat mustalaatikkotestaus ja valkolaatikkotestaus. Näitä lähestymistapoja käytetään pääasiassa kehitysvaiheessa.
(Glenford, Sandler & Badgett 2011, 8.)

Seuraavaksi esitellään muutamia yleisesti tunnettuja testausmenetelmiä ja kohdeyritykselle mielenkiintoisia testauksen lähestymistapoja. On olemassa myös muita testausmenetelmiä, mutta niitä ei tässä työssä käsitellä.

2.2.1 Musta laatikko -testaus

Yksi tärkeimmistä ja perinteisimmistä testausmuodoista on musta laatikko -testaus, joka tunnetaan myös nimellä tieto-ohjattuna tai syöte-tuloste-ohjatuttuna testauksena. Metodi kuvittelee ohjelman olevan ikään kuin musta laatikko. Tavoitteena on olla täysin välinpitämätön ohjelman sisäiselle toiminnalle ja rakenteelle. Sen sijaan tulisi keskittyä löytämään seikkoja, joissa ohjelma ei käyttäydy sen teknisten tietojen mukaisesti. Tässä lähestymistavassa testidata on johdettu yksinomaan ohjelman määrittelystä, eikä se ota huomioon kuinka ohjelman sisäisen rakenteen tulisi toimia.
Testaus toteutetaan antamalla ohjelmalle syötteitä ja katsomalla mitä sen jälkeen tapahtuu. Tällä tavalla kaikkien virheiden etsimiseksi olisi syötettävä kaikki mahdolliset syötteet ja vielä kaikilla mahdollisilla tavoilla. Mustalaatikkotestausta käytetään monesti ohjelman tarkistukseen ja viimeiseen hyväksyntään. Tätä testausmuotoa voidaan kuitenkin käyttää missä vaiheessa testausta hyvänsä. Tieto-ohjattu mustalaatikkotestaus on SAP-ympäristössä yleisesti käytetty menetelmä. Tämä on myös usein menetelmä, joka voidaan automatisoida. Testausmenetelmän tulokset ovat usein yksiselitteisiä, joko tulos on odotettu tai ei odotettu. (Helfen & Trauthwein 2011, 57; Glenford, Sandler & Badgett 2011, 8-9; Kasurinen 2013, 66.)

2.2.2 Valkolaatikkotestaus

Toinen merkittävä lähestymistapa testaukseen on valkolaatikko eli logiikkaohjattu testaus. Tämä lähestymistapa mahdollistaa ohjelman sisäisen rakenteen tutkimisen. Valkolaatikkotestauksessa testidata saadaan ohjelmien logiikan tutkimisen perusteella. Suomeksi tätä testausmenetelmää kutsutaan lasilaatikkotestaukseksi. Valkolaatikkotestausta käytetään yleensä vain kehitysvaiheessa (developer test, error elimination tai debugging process). Menetelmä täydentää musta laatikko -testausta kiinnittämällä huomion myös järjestelmän sisällä tapahtuviin toimintoihin, kun ohjelma nähdään ikään kuin lasilaatikkona. Menetelmässä testaajan tulee tuntea ohjelman logiikkaa ja itse ohjelmointia hyvin ja pystyä tarkastelemaan sitä lähdekooditasolla. Menetelmä mahdollistaa myös oppimisen ohjelmasta. Tämäkään menetelmä ei riitä havaitsemaan puuttuvia ominaisuuksia tai huonosti tehtyä vaatimusmäärittelyä, eikä se sen vuoksi voi yksinään olla laadunvarmennustestausta. (Glenford, Sandler & Badgett 2011, 10; Kasurinen 2013, 67-68.)

2.2.3 Harmaa laatikko -testaus

Testausmenetelmänä harmaa laatikko yhdistelee musta- ja valkolaatikkotestausta. Mallin ideana on yhdistää molempien menetelmien parhaat puolet. Käytännössä mallilla pyritään varmentamaan, että järjestelmä täyttää vaatimukset, ja että sen lähdekoodi on varmasti tarkastettu. Harmaa laatikko -testausta käytetään, kun järjestelmää ei voida testata valkolaatikon tavoin kokonaisvaltaisesti, mutta paikallisiin komponentteihin päästään käsiksi. Esimerkiksi verkkokaupan toimintaa voidaan testata ohjelmakoodin tasolle, mutta palvelinratkaisua tai maksujen hyväksymiseen käytettyä pankkirajapintaa ei mahdollisesti voida testata kuin mustana laatikkona. (Kasurinen 2013, 68.)

2.2.4 Regressiotestaus

Regressiotestaaminen ei ole erillinen testausmuoto, vaan yleistermi uudelleen testaamiselle. Puhutaan siis regressiotestauksesta, kun jotain toimivan järjestelmän osaa muutetaan, ja halutaan todentaa toiminta myös muutoksen jälkeen. Samasta asiasta voidaan puhua myös, kun kehitysversiosta halutaan todentaa kaikkien toimintojen oikeellisuus. Tärkein regressiotestauksen ominaisuus on varmistaa, etteivät jo kertaalleen korjatut

ongelmat esiinny komponenttiin tehtyjen muutosten jälkeen, ja ettei mitään uutta ole mennyt rikki. Menetelmän ajatuksena on, että virheet sijoittuvat useimmiten uusiin komponentteihin, tai näitä käyttäviin toimintoihin. Järjestelmään tulisi siis jokaisen muutoksen jälkeen suhtautua, kuin se olisi kokonaan uudistettu. Regressiotestauksessa suoritettavat perustestitapaukset ovat myös otollisia testausautomaatiolle, koska testitapauksia toistetaan projektin aikana useasti, mikä on eräs testausautomaation keskeisimpiä vaatimuksia. (Kasurinen 2013, 69-70.)

2.2.5 Tutkiva testaaminen

Tutkivassa testauksessa (explorative testing) pyritään nimensä mukaisesti etsimään ja löytämään ohjelman mahdollisia vikatiloja. Malli poikkeaa siis edellä mainituista esimerkeistä siinä, että tutkivassa testauksessa testaaminen ei perustu etukäteen dokumentoituihin testitapauksiin. Toisin kuin testitapauspohjaisissa menetelmissä, jotka on suunniteltu ennen testaamista, testaajia ei ohjeisteta, kuinka testaus tulisi suorittaa. Tutkivassa testauksessa testaajat hyödyntävät kokemustaan ja ymmärrystään testattavasta järjestelmästä. Menetelmässä perustuu rinnakkaiseen testien suunnitteluun, suorittamiseen ja oppimiseen. Testejä suunnitellaan testauksen aikana, ja saatua informaatiota käytetään hyödyksi uusien testien luomiseksi. Juha Itkosen tutkimuksissa selvisi, että menetelmällä parannettiin kustannustehokkuutta, kun tarkka etukäteissuunnittelu jätettiin pois. Kun ohjelmistotestaukseen osallistuu ihmisiä erilaisista ryhmistä, tutkiva testaus on toimiva tapa hyödyntää eri sovellusalueen osaamista testauksessa. Menetelmä hyödyntää testaajien taitoa ja kokemusta suorittaa luovia ja dokumentaation ulkopuolelle jääneitä testitapauksia. Tutkimusten mukaan tämä manuaalinen menetelmä on tehokas tapa täydentää testausautomaatiota. (Itkonen 2011; Kasurinen 2013, 74.)

2.3 Testauspolitiikka

Ohjelmistotestaus tarvitsee muiden projektien tapaan organisaation johdon tuen. Ylimmän johdon tulee ymmärtää testauksen tarkoitus, mutta myös riittävän testauksen avulla saavutettavat hyödyt. Testauspolitiikka määrittää ohjelmistolta vaaditun laatutason ja testaukselta toivotun tahtotilan. Se määrittelee yleisesti, kuka testaa ja miksi testataan ja on eräänlainen yleissuunnitelma eli testauksen iso kehys.

Politiikassa otetaan kantaa missioon ja tavoitteisiin, järjestelmäkehityksen ympäristöön, testausprosessiin, laatuominaisuuksiin, sovellusalueen riskeihin ja laatukysymyksiin.

Testauspolitiikka sisältää:

- Tavoitteet: mitä halutaan saavuttaa sekä noudatettavat periaatteet.
- Organisaatio: kuka testaa, kuka laatii testauspolitiikan ja kuka päättää testaukseen liittyvistä asioista.
- Prosessi: kuka määrittelee miten testausta tehdään.
- Testaajilta vaadittavat asiat.
- Standardit ja muut toimintatavat, joita testauksessa pitää noudattaa.
- Mittarit: miten testauksen laatua tai kustannustehokkuutta mitataan organisaatiossa.
- Miten testausta kehitetään organisaatiossa.

Testauspolitiikan tehtävä on antaa selkeä kuva päätöksenteosta ja vastuista sekä testauksen tavoitteista. (Kasurinen 2013, 110-111.)

2.4 Testaustrategia

Testausstrategia yhdistää tekniikat hyvin suunnitelluiksi kokonaisuuksiksi ja määrittelee miten testausta suoritetaan. Se ikään kuin määrittelee pelisäännöt testauksen tekemiselle. Tarvittavan testauksen määrää kun on vaikea arvioida etukäteen. Testausstrategia täydentää testauspolitiikkaa vastaamalla siihen, miten testaus tulee tehdä. Tämä on ensimmäinen testaustoimintaa laajemmin käsittelevä dokumentti, jonka laatimisesta huolehtivat testausta ymmärtävät henkilöt. Testausstrategian luojien on oltava perillä siitä, miten testaus tällä hetkellä suoritetaan, mitä testaustyö tarkoittaa sekä miten testaustoimintaa tulisi kehittää nykyisestä. Se on siis sopimus, joka vastaa kysymyksiin: mitä testataan, kuinka testataan, milloin testataan, kuka testaa ja miksi testataan.

Testausstrategia kaikessa yksinkertaisuudessaan on löytää kaikkein tärkeimmät viat mahdollisimman halvalla. Testauspolitiikasta ja testausstrategiasta luodaan yleisluontoiset ohjeet, jotka ovat helposti sovellettavissa kaikenlaiseen testaamiseen. Testausstrategia elää tarvittavan projektin mukaan, ja yrityksellä voi olla useita eri strategioita saman-

aikaisesti. ISO/IEC 29119 standardin mukaan strategiaan vaikuttavat aikaisempien projektien loppuraportit, laatuvaatimukset ja sisäinen laatumalli. Kattavassa testausstrategiassa määritellään seuraavat asiat:

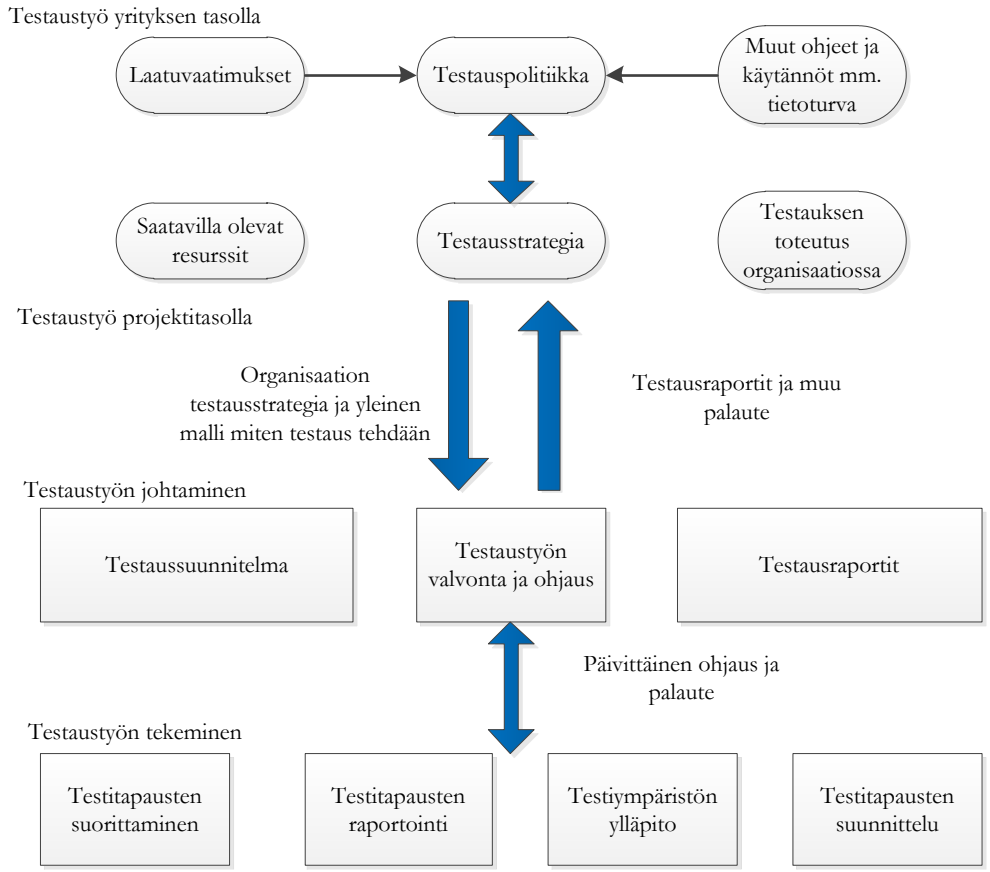
- Riskit: hallinnan toimenpiteet ja menetelmät.
- Ehdot: milloin testaus aloitetaan tai milloin ohjelma palautetaan korjattavaksi.
- Päätösvalta: mistä asioista testausta suorittavat henkilöt saavat päättää, ja kuka käyttää ylintä päätösvaltaa.
- Organisaatio: ketkä muodostavat testausorganisaation ja kuka on esimies, varsinkin jos muodostettu ryhmä on eri kuin testauspolitiikassa määritelty.
- Dokumentointi: mitä dokumentteja testauksessa käytetään, mitä niihin merkitään ja kuka niitä täyttää.
- Vaiheet: mitä työvaiheita testauksessa käydään läpi.
- Tekniikat: millä menetelmillä ja työkaluilla testausta suoritetaan.
- Testitapaukset: miten testattavat tapaukset valitaan ja priorisoidaan.
- Ympäristö: millainen järjestelmä testausta varten tarvitaan ja kuka siitä vastaa.
- Testaus: kriteerit joiden perusteella testataan uudelleen.
- Lopetusehdot: määritellyt ehdot milloin testaus voidaan päättää ja hyväksyä käyttöön otettavaksi.
- Poikkeamien hallinta: kuinka poikkeamia hallitaan, raportoidaan ja hoidetaan.

(Kasurinen 2013, 111-113.)

2.5 Testausprosessi

Ohjelmistotestauksen uusi kansainvälinen laatustandardi on ISO/IEC 29119, jonka tarkoitus on luoda kaikenlaisille testauksen lähestymistavoille sopiva pohja. Malli ei ota kantaa suoraan siihen, kuinka ohjelmisto tuotetaan tai minkäkokoisesta projektista on kysymys. Se määrittelee peruskomponentit, jotka jollain tapaa löytyvät jokaisesta testausta tekevästä organisaatiosta, eikä ota tarkemmin kantaa kuinka ohjelmisto tulisi tuottaa. Malli on tämän vuoksi hyvä koko organisaatiossa tapahtuvan testauksen ymmärtämiseen ja opettamiseen. Yrityksen näkökulmasta ISO/IEC 29119 on hyödyllinen myös sen vuoksi, että sen toimintaa voidaan testata auditoinneilla, tai todentaa

standardia hankkimalla sitä koskeva sertifikaatti. Lisäksi tästä kansainvälisestä standardista on saatavilla runsaasti lisätietoa eri kielillä. (Kasurinen 2013, 103.)



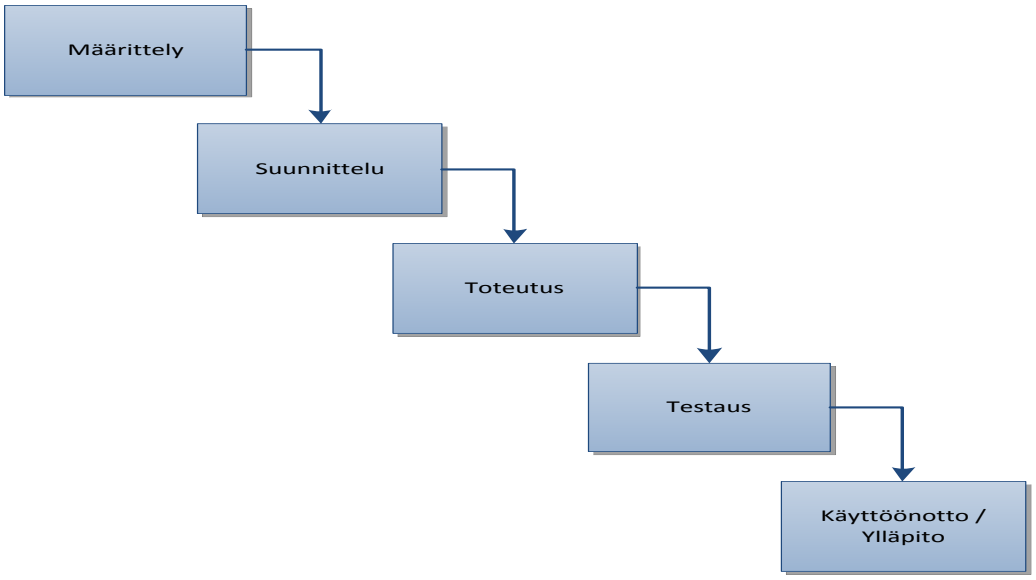
Kuvio 3. Testauksen yleinen toimintamalli ISO/IEC 29119- mallin periaatteita noudattaen: organisaation taso ohjaa projektitasoa, jolla testausta tehdään. (Kasurinen 2013, 103.)

Myös globaalisti tunnustettu prosessikehys, ITIL (The Information Technology Infrastucture Library) IT-palveluiden hallintaan ja johtamiseen, sisältää kuvauksen testauksen prosesseista. ITIL:issä testaus on osa palvelun siirtoa (Service Transition). ITIL:in testausprosesseissa ei käytetä nimitystä hyväksymistestaus, vaan palvelujen hyväksymistestaus. Palvelun hyväksymistestaus alkaa vaatimusten todentamisella. Palvelun hyväksymiskriteerit ja –testaussuunnitelman hyväksyvät muutosten takana olevat henkilöt, hyvin usein siis liiketoiminta. Tämän mallin kuvaaman testausprosessiin kuuluu 7 eri vaihetta:

- Validointi ja testauksen hallinta (Validation and test management).

- Testauksen suunnittelu ja määrittely (Plan and design).
- Testaussuunnitelman ja määrittelyjen tarkastus (Verification of test plan and design).
- Testiympäristön valmistelut (Preparation of the test environment).
- Testauksen suorittaminen (Testing).
- Lopetusehtojen arviointi ja raportointi (Evaluate exit criteria and report).
- Testiympäristöjen siivous sekä testauksen päättäminen (Clean up and Closure). (Office of Goverment Commerce 2007, 122-134.)

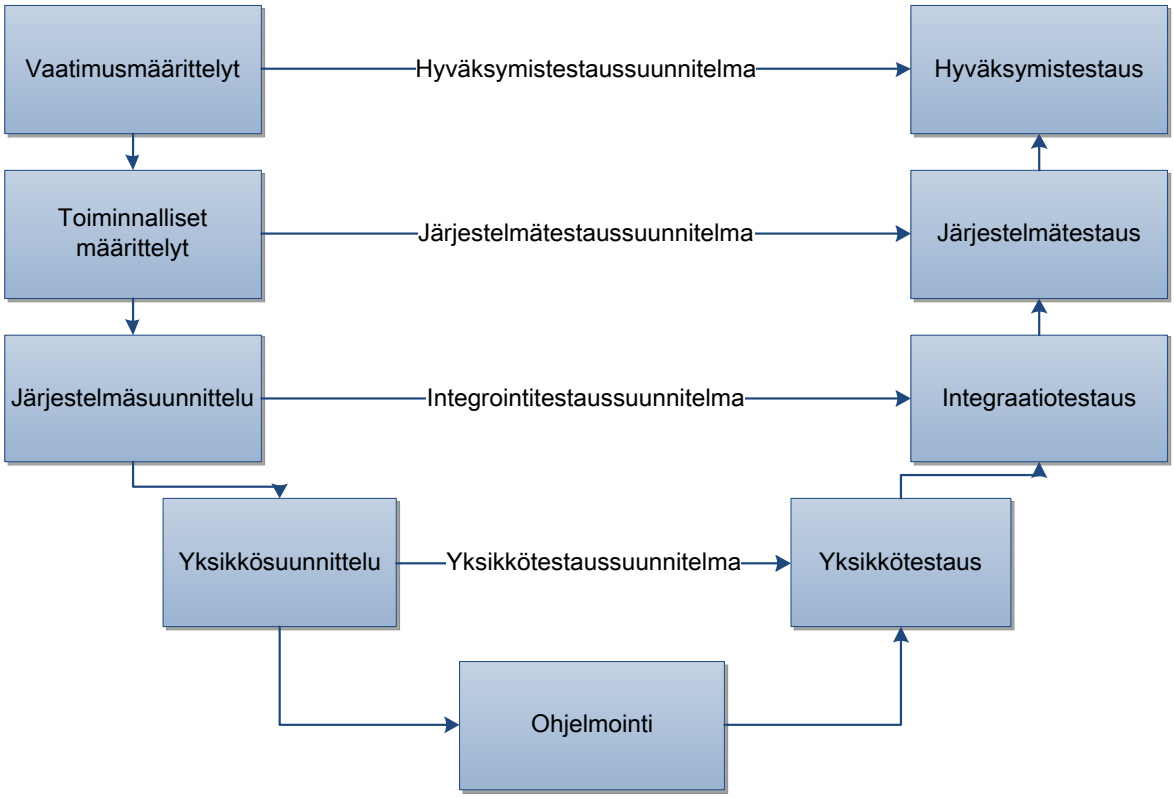
Yleinen mielikuva testauksesta ja sen toteutuksesta on se, että testaus on vain yksi työvaihe perinteisessä ohjelmistotuotannon vesiputousmallissa. Projekti etenee tässä mallissa vaihe kerrallaan alkaen vaatimusten määrittelystä. Testaus on siis ainoastaan yksi työvaihe toteutuksen ja käyttöönoton välissä. Malli on monella tapaa vanhanaikainen, sillä esimerkiksi vaatimusten täydellinen määrittäminen ja kerääminen etukäteen on monissa projekteissa mahdotonta. Lisäksi testausta tehdään vain yhdessä vaiheessa, johon päästäessä suunnittelu- ja kehitystyö on suurilta osin tehty. Parempi malli ohjelmistojen kehitystyölle on seuraavaksi esiteltävä V-malli. (Kasurinen 2013, 13-14.)



Kuvio 4. Ohjelmistotuotannon vesiputousmalli (Kasurinen 2013, 13.)

2.6 Testaustasot ja V-malli

Yleinen V-malli kuvaa suoraa ohjelmistokehitysprosessia. Malli sopii niin vesiputousmallilla kuin inkrementaalisilla menetelmillä toteutettuihin ohjelmistoprojekteihin. Malli kuvaa hyvin sitä, kuinka eri määrittelyvaiheet liittyvät testauksen tasoihin. Yleensä testivaiheet seuraavat yleistä V-mallia. Boehm esitteli tämän mallin jo vuonna 1979. Vaikka malli on vanha, sitä voidaan edelleen käyttää kuvaamaan ohjelmistotestauksen eri tasoja myös ketterien kehitysmenetelmien kanssa. Malli ei sellaisenaan ole riittävä ohjelmistojen käyttöönottoon ja kaikkeen kehitystyössä tapahtuvaan toimintaan, mutta se kuvaa kuitenkin hyvin mitä vaiheita kehitysprosessiin kuuluu ja miten ne liittyvät toisiinsa. Mallia voidaan käyttää myös SAP-ympäristössä tapahtuviin kehitystöihin, käyttöönottoihin tai päivityksiin. (Helfen & Trauthwein 2011, 49.)



Kuva 1. Ohjelmistotestauksen viisitasoinen V-malli, jossa muutoksena Kasurisen malliin on lisätty yksikkösuunnittelu ja viety ohjelmointi omaksi tehtäväksi. Näin toimitaan kohdeyrityksessä. (Kasurinen 2013, 14.)

V-malli kuvaa suunnittelua rakentamisvaiheessa ja testaustasoja testausvaiheessa. Malli määrittelee tekniselle testaukselle kolme eri vaihetta: yksikkötestauksen, integraatiotes-

tauksen ja järjestelmätestauksen. Näitä nimitetään testaustasoiksi, koska testausta suoritetaan eri tasoilla. Yksikkötestauksessa testaus kohdistetaan yhteen komponenttiin, integraatiotestauksessa muutamaan komponenttiin ja näiden väliseen toimintaan. Järjestelmätestauksessa testataan jo kokonaisuuden toimintaa suorituskyvyn ja toiminnallisuuden osalta, mutta tämä tapahtuu vielä testiympäristössä. Hyväksymistestausta voidaan tehdä joko kohdeympäristössä tai sen tarkassa simulaatiossa. Perusteellinen ero SAP-ympäristössä tapahtuvaan ohjelmistotuotantoon on taustalla oleva (The ASAP implementation roadmaps) prosessimalli. (Helfen & Trauthwein 2011, 52-53; Kasurinen 2013, 50-51.)

2.7 Hyväksymistestaus

Ohjelmistotestauksen vaiheita kuvaavan V-mallin viimeinen vaihe on hyväksymistestaus. Hyväksymisvaiheessa ohjelmaa testataan tilaajan kanssa. Vaiheen tarkoitus on todentaa ohjelman olevan valmis täyttämään sille vaatimusmäärittelyssä asetetut tavoitteet. Hyväksymistestaus (Acceptance Testing, tai User Acceptance Testing eli UAT) on testauksessa se vaihe, jossa järjestelmä virallisesti tarkastetaan. Vaiheen onnistuneen läpiviennin seurauksena asiakas hyväksyy tuotteen onnistuneeksi ja ohjelmisto siirtyy lakiteknisestikin asiakkaan omaisuudeksi. Hyväksymistestauksessa järjestelmää käytetään monesti sen virallisessa kohdeympäristössä. Hyväksymistestauksen tarkoituksena on tarkastaa ja varmentaa, että sille määritellyt ominaisuudet on täytetty. Tämä tarkastelu tehdään tilaajan tai loppukäyttäjän näkökulmasta. Hyväksymistestaus on aina tilaajan vastuulla. (Helfen & Trauthwein 2011, 49-51.)

Standardeja, kuten ISO/IEC 25000:ta, käytetään ohjelmiston hyväksymiskriteereiksi ja niiden määrittelyyn. Lisäksi amerikkalainen CISQ (Consortium for IT Software Quality) on määritellyt ohjelmistotuotteille viisi yleistä laatuattribuuttia. Ne ovat riittävän yleisluontoisia, jotta ne kuuluvat jokaiseen valmistettavaan ohjelmistotuotteeseen. Nämä ohjelmistosta testattavat ominaisuudet ovat:

- Security – turvallisuus
    - Tarkoittaa järjestelmän kykyä vastustaa hyökkäyksiä tai murtautumisyrityksiä.

- Reliability – luotettavuus
    - tarkoittaa ohjelman kykyä toimia ilman merkittäviä katkoksia, ongelmia tai riskitekijöitä.
- Efficiency – tehokkuus
    - Tarkoittaa ohjelman kykyä suorittaa tehtävänsä mahdollisimman lyhyessä ajassa.
- Size – koko
    - Tarkoittaa sitä, että ohjelman tulee olla rakenteeltaan oikeaa kokoluokkaa.
- Maintainability – ylläpidettävyys
    - Tarkoittaa ohjelman siirrettävyyttä alustalta toiselle ja mukautuvuutta uusiin tapauksiin ja käyttötilanteisiin.

(Kasurinen 2013, 137.)

Kun ohjelmistolta testataan vain sen toiminnallisuutta, käytetään testitavasta nimitystä functional testing eli toiminnallinen testaus. Tapa nimensä mukaisesti tutkii vain ohjelman toimivuutta, eikä ota kantaa sen suorituskykyyn. Performance testing eli suorituskykytestaus taas tutkii, toimiiko ohjelma tarpeeksi tehokkaasti.

Testitapa	**Testitavoite**	**Testattava ominaisuus**
Toiminnallisuustestaus	Saadaan odotettu tulos	Oikeellisuus
Suorituskykytestaus	Saadaan odotettu tulos odotetussa ajassa	Toiminnallinen tehokkuus

Taulukko 1. Toiminnallisuustestauksen ja suorituskykytestauksen erot. (Helfen & Trauthwein, 47.)

2.8 Laadun varmistaminen

Ohjelmistotestauksen tehtävä on siis varmistaa tuotoksen laadukkuus ja vaatimustenmukaisuus. Laadulla tarkoitetaan tässä yhteydessä sitä, että tulokset täyttävät odotukset. Laadulla voidaan kuitenkin tarkoittaa teknistä laatua, asiakkaan kokemaa laatua, tai tuotteen hinnan perusteella odotettua laatua. Ohjelmistoyritysten mukaan testausta tärkeämpi laadun lähde on suunnittelu- ja kehitystyö. Huonosti suunniteltua

tai toteutettua ohjelmistoa kun ei saa testauksella korjattua. Testauksen avulla voidaan osoittaa ohjelmassa olevan virheitä, mutta ei täydellistä virheettömyyttä. Parhaiten virheitä löytää ulkopuolinen, ohjelmiston kehitykseen osallistumaton henkilö. On hyvä muistaa, että testaus ei yksinään riitä laadunvarmistamiseen, vaan se on osa laadunhallintaa kokonaisuutena. Mitä ohjelmistoista sitten pyritään testauksella ja laadunvarmistuksella etsimään. ISTQB:n testaussanastossa ongelmat määritellään seuraavasti:

Virhe tai erehdys (mistake) on käyttäjän toimintaa, joka tuottaa väärän tuloksen.

Vika tai bugi (error tai bug) Komponentissa, järjestelmässä tai dokumentaatiossa oleva poikkeama, joka voi estää ohjelmaa toimimasta vaaditulla tavalla.

Häiriö (failure) Ohjelmiston toiminta poikkeaa odotetusta. Tilanteen voi aiheuttaa ohjelmistovian lisäksi lähes mikä tahansa asia, joka estää järjestelmää toimimasta kunnolla.

Näiden eroavuudet on hyvä käsitellä, sillä näitä nimityksiä käytetään usein toistensa synonyymeinä. (Kasurinen 2013, 50.)

Projektin tiukka aikataulu saatetaan kiriä umpeen kunnollisen testauksen laiminlyönnillä. Ohjelmistotestauksessa virheen lopulliseksi hinnaksi tulee etsimisestä ja testauksesta aiheutuvat kulut: virheen hinta = virheen etsimisen hinta + virheen korjaamisen hinta + uudelleen testaamisen hinta. (Helfen & Trauthwein 2011, 381.)

Laadulla voidaan tarkoittaa esimerkiksi tuotteen teknistä laatua tai asiakkaan kokemaa laatua. Ohjelmistoprojektin tavoitteena on periaatteessa aina pyrkiä parhaaseen mahdolliseen laatuun. Tämä tehdään toteuttamalla ensin kehitystyöllä paras mahdollinen järjestelmä ja sitten testaustyöllä varmistaen, että järjestelmä täyttää määrityksen ja laatuvaatimukset. Laadun hinnaksi voidaan laskea yhteishinta, jonka muodostavat virheiden estämiseksi tehty testaus ja tuotteeseen jääneet virheet. (Kasurinen 2013, 132-133.)

3 Haastattelut ja nykytila-analyysi

Tässä luvussa kerrotaan tehdystä kvalitatiivisesta eli laadullisesta tutkimuksesta. Tutkimustapana käytettiin teemahaastattelua. Tutkimuksen kohteena oli yritys X:n nykyinen sovellustestausprosessi. Tutkimuksen lähestymistavalla varmistettiin, että tuloksiin saatiin testaustyötä tekevien näkökulma nykytilaan. Haastatteluilla haettiin myös näkemyksiä siihen, kuinka sovellustestausta tulisi jatkossa tehdä, ja miten sitä tulisi hallinnoida. Tutkimukseen valitut asiantuntijat vastaavat kukin oman järjestelmäkokonaisuuden toiminnasta, kehittämisestä ja testaamisesta. Mukana olevat tietohallintopäälliköt vastaavat oman ryhmänsä toiminnasta ja kehittämisestä.

Haastatteluihin valitut henkilöt ovat työnohjaajan lisäksi viisi muuta yritys X:ssä testausta tekevää asiantuntijaa ja tietohallintopäällikköä. Haastattelut suoritettiin 23.3.2015-1.4.2015. Toimeksiantajan edustaja valitsi haastateltavat henkilöt. Tutkimuksessa haluttiin laadukkaita vastauksia tutkimusongelmiin. Lisäksi tutkittiin minkälaisena haastateltavat kokevat sovellustestausprosessin henkilökohtaisesti. Heille toimitettiin haastattelukutsujen yhteydessä teemahaastattelun runkokysymykset, joten heillä oli aikaa pohtia varsinaisia kysymyksiä ennen haastattelua. Varsinkin testauspolitiikkaa ja testausstrategiaa käsittelevät kysymykset nähtiin hyvin erilaisina. Kaikki haastattelut nauhoitettiin laadun varmistamiseksi ja haastattelun yhteydessä tehtiin myös muistiinpanoja paperille. Jokainen haastattelu litteroitiin eli purettiin tekstimuotoon nauhoituksesta. Tämän jälkeen se kirjoitettiin paremmin luettavaan muotoon lukijaa silmällä pitäen. Haastattelussa esitetyt kysymykset, joihin näillä etsitään vastauksia:

Miten nykyiset testisuunnitelmat toimivat?

Miten nykyiset testitapaukset tyypillisesti toimivat?

Mitä koet nykyisessä testausmallissa toimivaksi?

Minkälaisia puutteita tunnistat nykyisessä testausmallissa?

- Kysymykset vastaavat tutkimusongelmiin: Mikä on yritys X:n sovellustestauksen nykytila?

Kuinka yritys X:n nykyinen testausprosessi toimii?

Kuinka nykyiset testilomakkeet mielestäsi toimivat?

Mitä ajattelet nykyisistä testaustyökaluista?

Minkälainen on nykyinen dokumentointikäytäntö?

- Kysymykset vastaavat tutkimusongelmaan: Onko testaustyötä tekevillä käytössään tarvittavat työkalut?

Minkälainen on yritys X:n nykyinen testauspolitiikka?

Minkälainen on tämän hetkinen testausstrategia?

- Kysymykset vastaavat tutkimusongelmaan: Tunnistetaanko nykyinen testauspolitiikka tai – strategia tietohallinnossa?

Miten testisuunnitelmaa tulisi muuttaa?

Miten testitapauksia tulisi muuttaa?

Miten testaus tulisi jatkossa tehdä?

Mitä nykyisestä testausmallista tulisi säilyttää myös uudessa mallissa?

Minkälaisena näkisit uudet testilomakkeet?

Minkälaisia testauksen työkaluja näkisit jatkossa tarvittavan?

Miten dokumentointikäytännön tulisi jatkossa kulkea?

- Kysymykset vastaavat tutkimusongelmaan: Miltä uusi käytännön testausta tukeva prosessi ja aktiviteettiluettelo voisi näyttää?

Minkälaisena näkisit uuden testauspolitiikan?

Minkälaisena näkisit uuden testausstrategian?

- Kysymykset vastaavat tutkimusongelmaan: Miltä uusi testausstrategia voisi näyttää?

3.1 Teemahaastattelut

Teemahaastattelussa kysymykset jaettiin kahteen osaan. Ensimmäisen osan tarkoitus oli selvittää nykytilaa, ja toisen osan kysymykset selvittivät miten sovellustestausprosessin tulisi jatkossa kulkea. Teemahaastattelun kysymykset ovat liitteessä 1. Haastattelussa mukana olleet henkilöt:

H1: Tietohallintopäällikkö, testauskokemusta 16 vuotta.
H2: Tietohallintopäällikkö, testauskokemusta 15 vuotta.
H3: Järjestelmäpäällikkö, testauskokemusta seitsemän vuotta.
H4: Järjestelmäasiantuntija, testauskokemusta 12 vuotta.
H5: Järjestelmäpäällikkö, testauskokemusta kolme vuotta.
H6: Tietohallintopäällikkö, testauskokemusta neljä vuotta.

3.2 Testauksen nykytila

Yritys X:n sovellustestauksen nykytilaa selvittävillä kysymyksillä oli tarkoitus kartoittaa testauksen ongelmakohtia. Lisäksi tiedusteltiin käytössä olevista testaustyökaluista sekä käytössä olevasta testausstrategiasta.

3.2.1 Kysymys ”Mikä on yritys X:n sovellustestauksen nykytila?”

Tähän kysymykseen saatiin hyvin samanlaisia vastauksia kaikilta haastatelluilta. Esimerkiksi H1:n mielestä testisuunnitelmat ja -tapaukset toimivat vaihtelevasti. Moduulitestaus on hyvinkin kattavaa, sillä tämän osa-alueen asiantuntija keskittyy pariin hyvin tuntemaansa moduuliin. Tämä osa-alue toimii hyvin. Kun siirrytään pitkään prosessiin ja regressiotestaukseen, eri variaatiot eivät tule kunnolla testatuksi. Esimerkiksi SAPin EhP-päivityksissä (SAP Enhancement Package) pitkä prosessi pitäisi testata todella kattavasti. H1 kuvaileekin testauksen nykyistä tilaa sietämättömäksi. H1: ”Vaikka EhP-päivityksissä testataan viikkoja, ei nykyisillä testityökaluilla voida testata kaikkea tarpeellista”. Nykyiset testisuunnitelmat ovat hyvin yksinkertaistettuja. 80–20 -sääntö toimii käytännössä, kun tehdään suunnitellut asiat. Ohjelmista pyritään siis testauksella löytämään vakavat virheet ja 80 prosenttia muista virheistä. Pitkän prosessin testauksessa haasteena on henkilöosaaminen: ei pystytä riittävän itsenäiseen tekemisen. Testauksen

jaksottaminen on myös haastavaa, kun eri alueiden osaajia tarvitaan samaan aikaan monessa eri testauksessa. H1 kertoo, että viimeisen osuuden tekijällä tulee usein kiire, koska aikataulu kiritään umpeen, ja prosessin loppupäässä tehtävästä testauksesta karsitaan. Tästä muodostuu nykyisen testauksen pullonkaula.

H1 väittää, että nykymallin toimivuus johtuu pelkästään osaavista testaajista. He tietävät, mitä regressiotestauksessa pitää ottaa huomioon. Hänen mielestään testausprosessi on liian henkilöitynyt tällä hetkellä. Testausta tekevät siis osaavat henkilöt, eikä kyse ole muuten toimivasta mallista. Nykyinen malli toimii niin kauan kun samat osaavat henkilöt ovat testaustyön käytettävissä. H1 osallistuu testaukseen paljon ja näkee testauksen tilan melko huolestuttavana. Hänen mielestään osaamisen jakamisessa ja varmistamisessa on paljon tekemistä – nyt ja jatkossa.

H2:n mukaan testaus on jokaisen asiantuntijan ja järjestelmävastuullisen omalla vastuulla. Periaatteet testaukseen ovat olemassa, mutta vastuullinen suunnittelee itse, mitä tulee testata. Eri asia on, osaako hän suunnitella tarpeeksi hyvin. H2: ”kaikilla ei ole aina sadan prosentin tietämystä siitä, mitä juuri tämä kehitystyö vaatii testaukselta. Sen vuoksi saatetaan testata myös liikaa ja turhia asioita”. H2 kertoo, että kunkin eri osa-alueen asiantuntijalla on päävastuu omasta järjestelmästään ja siitä miten sen kehitystyötä sekä testausta tehdään. Nykymallissa suurimpina puutteina hän näkee laadun varmistamisen mittarien puuttumisen. Kun selkeät mittarit puuttuvat, ei tiedetä minkälaista jälkeä järjestelmätoimittajat tuottavat, eikä pystytä kertomaan, kuinka paljon virheitä menee testauksesta läpi tuotantoon asti. Mittareiden puuttuminen vaikuttaa myös kokonaiskuvan muodostamiseen.

H2 kertoo oman ryhmänsä asiantuntijoiden käyttävän hyväksi liiketoiminnan asiantuntemusta testitapausten suunnittelussa. H2: ”Kehitystyön tilaaja ja tietohallinnon vastuullinen asiantuntija miettivät yhdessä, millaisia testitapauksia työ vaatii. Hyvään testaukseen kuuluu muutakin kuin kehitettävissä olevan toiminnon testaus”. Tulisi siis testata aina myös muut toiminnot, kun tehdään uusia kehitystöitä. H2 mukaan monissa kehitystöissä vaaditaan jo niin paljon tietämystä liiketoimintaprosessien toiminnasta, että ilman liiketoiminnan apua testitapauksia ei edes voitaisi suunnitella.

H3 sanoo testauksen olevan 80 prosenttisesti manuaalista tekemistä. Tämän vuoksi tekemisessä korostuu, kuka testausta on tekemässä. Testauksen laatu eli kattavuus ja löydettyjen virheiden määrä on henkilöriippuvaista. Malli siis toimii hyvin, jos osaava henkilö on testaamassa.

3.2.2 Kysymys ”Kuinka yritys X:n nykyinen testausprosessi toimii?”

H1 sanoo käytännön toimivan vaihtelevasti: ”osa kirjaa dokumentteihin vain testitapauksen numeron eli testi-id:n, joka on voimassa, kunnes testijärjestelmä tuoreistetaan vastaamaan tuotantoympäristöä. Testipöytäkirjan tapaukset katoavat tuoreistuksessa, kun on kirjattu vain testi-id”. H1 näkee tämän esteenä oppimiselle. Huonona puolena tässä on myös toistettavuuden puuttuminen. H1 sanoo, että testausympäristössä on ylimääräisiä töitä, joita ei koskaan siirretä tuotantoon. Testiympäristön siivous pitäisi tehdä useammin ja saada ylimääräiset työt sieltä pois. Hänen mukaansa testiympäristö ei toimi kunnolla missään ympäristössä, kun kehityksiä aloitetaan, mutta ei päätetä. H1 toteaa, että ”tehdään paljon turhaa koodia toimittajan puolelta, eikä versionhallintakaan aina toimi. Tämä saattaa johtaa siihen, että viedään vahingossa vanhoja tapauksia tuotantoon”.

H2 kertoo, että iso tai pieni kehitystyö, niin dokumentointi vie yhtä paljon aikaa. Yhden kehitystyön dokumentointi vie 30–60 minuuttia asiantuntijasta riippuen. H2:n mukaan tarvittavat asiat tulee testattua pääpiirteissään, mutta joskus virheet huomataan vasta tuotannossa. H2 sanoo, että ”toimivinta ovat asiantuntijat. Nykymallissa vasta UAT:ssa liiketoiminta on varmasti mukana testauksessa. Täydentämisen ryhmässä liiketoiminta on mukana jo moduulitestauksessa. Näin varmistetaan, että kehitystyö on varmasti sellaista, mitä liiketoiminta on tavoitellut”. H2 muistuttaa, että virheet pitää löytää ja poistaa mahdollisimman aikaisin. Hän sanoo myös osan kehitystyöstä olevan jo niin haastavaa, että liiketoiminnan on pakko olla mukana testauksessa alusta lähtien. H2: ”Tietohallinnossa järjestelmäosaaminen kyllä riittää, mutta kokonaisprosessin ymmärtämiseksi tarvitaan liiketoimintaa”. Hän sanoo tämän mallin toimineen heillä hyvin.

H2 totesi, että testilomakkeet ja -dokumentit toimivat ihan hyvin. Käytössä ovat vakiolomakkeet, joiden täyttämisessä on kuitenkin eroja. Toiset kuvailevat testitapaukset erittäin tarkasti kuvineen ja toiset taas hyvin suppeasti. Useasti vaikeassa paikassa dokumentoidaan kuitenkin hyvin kattavasti. Varsinkin, kun kyseessä on hyvin monimutkainen prosessi. Yksikkötestaus on hänen mukaansa yleisesti tarkimmin dokumentoitu, ja hyväksymistestauksessa dokumentointi tai sen kuvaus on kevyempää.

Testivaiheesta pitäisi löytyä dokumentit, jotka nykyään kirjataan erilliseen sovellukseen. Kaikki muut määritykset ja testidokumentit tallennetaan intranetiin, kertoo H2. Kun tallennetaan kahteen eri paikkaan, tehdään turhaan ylläpitoja. Molemmat tallennusjärjestelmät ovat käytössä myös IT:n sisäisessä keskustelussa, ja näin tehdään tuplasti työtä. Tämä tapa toimii kuitenkin melko hyvin. H2: ”Ei vain ole yhtenäistä mallia, vaan jokainen tapaus käydään omanaan läpi, ja tämä perustaa hyvin pitkälle asiantuntijuuteen”.

H3 näkee ongelmaksi manuaalisen testausmäärän. Kun pitää testata manuaalisesti ja on käytävä läpi kaikki prosessit, joita on tuhansia. Hänen mukaan testaus tehdään usein suurpiirteisesti. SAPin versionvaihdoissa tai muissa raskaissa toimenpiteissä testaukseen sidotaan kymmenen henkilöä kolmeksi viikoksi. H3 katsoi nykyisen mallin parhaaksi puoleksi joustavuuden ja katsoo sen olevan säilyttämisen arvoinen ominaisuus myös jatkossa. H3 sanoo, että joskus testisuunnitelmaa ei vain ole. Suunnitelmat eivät ole menneet aikataulun mukaisesti yhdessäkään projektissa, missä H3 on ollut mukana. Aikataulu on venynyt aina. Kun pienkehitystä tehdään paketteina jopa kahdeksan kertaa vuodessa, testaukseen suunnitellaan vain viikko aikaa. Ei siis tehdä riittävän kattavasti testausta. Pienkehityksessä vain aikataululla on merkitystä. Nykyisessä mallissa liiketoiminta voi myös jättää töitä määräajan jälkeen, joka taas näkyy testauksessa. Näin hävitään useita päiviä loppupään työstä, sanoo H3.

Myös H3:n mukaan testisuunnitelmat toimivat vaihtelevasti. Riippuen suunnitelman tekevästä henkilöstä ja kyseessä olevasta kokonaisuudesta. Testisuunnitelman toteutuminen riippuu täysin henkilön osaamisesta, kuvailee H3. Hänen mukaansa aikataulu ei yleensä pidä testaussuunnitelmien mukaisesti, eikä testisuunnitelmia aina edes tehdä. Pienkehityksen osalta pakettiaikataulu pitää kuitenkin huolen aikataulusta. Itse

testaus tehdään asiantuntijan toimesta ja ilman suurempaa etukäteissuunnittelua. Tämä siksi, että pienkehityksen osalta viemme useita satoja muutoksia tuotantoon vuodessa. Asiantuntija tekee testitapaukset, joilla pyritään kattamaan testaus niin, että tärkeimmät toiminnallisuudet saadaan varmistettua halutusta muutoksesta tai korjauksesta. Tämä myös mahdollistaa joustavuuden ja mahdollisuuden tehdä tarkkoja testauksia riippuen testaustarpeista ja niiden testauskriittisyydestä. Isoissa muutoksissa tai version vaihdoissa testausmäärä on suuri, kun testaus toteutetaan manuaalisesti. Prosessien valtavan määrän vuoksi ei kaikkia pystytä testaamaan. Testaus ei siis ole täysin kattavaa, H3 painottaa.

H3 kertoi, että nykyisellään dokumentointi on aika raskas prosessi, mutta osittain pitää tehdä, jotta pysytään kartalla siitä, mitä tehdään. Hänen mukaan nykyiset dokumenttien säilytyspaikat tulee saada yhdistettyä – eli toivottavasti uusi järjestelmä ajaa tämän asian ja tämä menee parempaan suuntaan. Nykyisen dokumentointikäytännön H3 katsoo toimivan suhteellisen hyvin, mutta nykyisessä mallissa kentät ovat aika pieniä ja näin ollen esimerkiksi kuvankaappaukset jäävät aika pieniksi. Nykyinen malli on pystyssä oleva dokumentti, mutta H3 on pohtinut, olisiko landscape –malli parempi. Lisäksi hän katsoo testilomakkeessa olevan saman kentän kahteen kertaan ja toivoo selkeämpää dokumenttipohjaa.

H3: ”Pienkehityksessä tehdään toimittajalle testitapaus, jonka perusteella toimittaja rakentaa ohjelman. Kesken testauksen tehdään myös lennosta (jos osataan) erilaisia kokeiluja. Testaussuunnitelmassa ei oteta huomioon tätä ad-hoc -testauksen tai tutkivan testauksen vaatimaa lisäaikaa”.

H4 kertoo testitapausten toimivan kohtuullisesti. Kuitenkin siihen välillisesti liittyvät asiat jäävät usein testaamatta tai ne testataan huonosti. Nykymallissa testataan moneen kertaan, kun ensin toimittaja testaa, sitten IT testaa, jonka jälkeen IT testaa yhdessä liiketoiminnan kanssa. Testaus moneen kertaan on toimiva (toimittaja, IT ja liiketoiminta). Testaussuunnitelmissa ei aina tunnisteta, mihin kaikkeen muutos vaikuttaa, toteaa H4. Hän kokee suurimmat puutteet kokonaisuuksien testauksessa. Kehityspaketeissa on paljon muutoksia, ja yleistestaus eli regressiotestausautomaatio puuttuu, toteaa H4. SAPin testausympäristöt pakottavat tietyn mallin käyttämiseen. H4 kokee

testilomakkeiden kielen kanssa epäselvyyksiä, kun käytetään liiketoiminnan kanssa suomea ja järjestelmätoimittajan kanssa englantia. Hän toivoisi tähän selkeämmät ohjeet ja universaalin kielen testaukselle.

H5 sanoo, ettei tällä hetkellä ole täysin luotettavaa keinoa todeta, että testattaisiin kaikkia tarvittavia asioita. Pienkehityksessä luotetaan vain toimittajan sanaan siitä, mitä tulee testata ja mitä ei. Yhtenäinen malli puuttuu tai tällaista ei ainakaan ole kirjattu. H5 pitää myös tärkeänä, että testitapaukset vahvistettaisiin jollain siihen tarkoitetulla työkalulla. Nykyisestä mallista tulisi säilyttää testauksen kulku, hän toteaa. Lisäksi hän toivoo, että kuvakaappausten lisääminen lomakkeisiin tehtäisiin helpommaksi ja peräänkuuluttaa testaushallintatyökalun välttämättömyyttä. H5 kertoo, että manuaalinen testaus pitää saada ensin tarpeeksi hyvälle tasolle, jotta voidaan arvioida automatisointia manuaalitestauksen tueksi. Sekä H5 että H6 olivat yksimielisiä siitä, että dokumentointi tulee upottaa mukaan muutoksen hallintaan.

H5:n mukaan ongelmia on ollut kohtuullisen vähän, vaikka paljon kehitetäänkin. Hänen mukaansa vain vähän ongelmia ilmenee tuotantoon viennin jälkeen. Hän sanoo, että nykyisessä mallissa asiat tulevat testatuksi moneen kertaan ja myös muutokset testataan. Hän jatkaa kuitenkin, että testataan varmasti myös vääriä asioita tällä hetkellä.

H6 kertoo, että SAP-kehityksessä testisuunnitelmat ovat korkealla tasolla. Tekeminen vaatii kuitenkin paljon käsityötä. Hänen mukaansa tekemisen taso vaihtelee alueesta ja kehityksestä riippuen. Joihinkin SAPin toimintoihin on olemassa testikirjasto, mutta CATMAN –osa-alueella ei ole olemassa samanlaista. H6:n mukaan viitseliäisyydestä johtuen liiketoiminta ei ole tehnyt kattavia testitapauksia. Muutenkin testitapaukset toimivat hänen mukaansa vaihtelevasti, eikä kunnon työkaluja testaukseen ole. Myöskään testauspolitiikkaa tai -strategiaa hän ei muista nähneensä. H6 kertoo, että kaikki testataan mitä pystytään, myös asiakas osallistuu testaukseen. Nykymallissa asiakas asettaa aikatauluja – ei muuta.

H6:n mukaan nykyinen testausmalli ei ole kehittynyt, järjestelmässä valmiina olevat tapaukset toimivat vain versiovaihdossa. Nykyinen testausmalli on myös epäselvä. Testitapaukset ovat puutteellisia, kun vajavaisen suunnittelun seurauksena toimittajalla

ja liiketoiminnalla on eri testitapaukset. Hänen mukaan toimittajalle pitäisi saada kattavat liiketoiminnan luomat testitapaukset. Yritys X:n toimesta tehdään tällä hetkellä liikaa testausta. Testausta voitaisiin teettää enemmän toimittajalla, toteaa H6, ja jatkaa: ”jos saisimme toimittajalle enemmän tietoa ja paremmin kattavat testitapaukset, pääsisimme itse vähemmällä testauksella”. H6:n mukaan nykyisellään toimittajalla ei ole tarpeeksi tietoa, eikä mallissa ole muutenkaan kehittyneitä ominaisuuksia, kun aika paljon tehdään Excelillä.

3.2.3 Kysymys ”Tunnistetaanko nykyinen testauspolitiikka tai -strategia tietohallinnossa?”

Tämä kysymys osoittautui haastavaksi kaikille vastaajille. Kukaan haastatelluista ei tunnistanut yritys X:n testauspolitiikkaa tai yleistä testausstrategiaa. H3 mainitsi testausstrategiaksi sen, että testataan kaikki mitä pystytään. Muuta politiikkaa tai testausstrategiaa hänkään ei tunnistanut. H3: ”Ei meillä taida tällaista olla, tai minä en ainakaan tätä sitten muista.” Muutama henkilö mainitsi haastattelussa 80–20 –säännön testauksen suorittamiselle ja virheiden havaitsemiselle. H2 ymmärtää testauspolitiikan nelivaihemallina: toimittajan testaus, IT:n moduuli-, integraatio ja regressiotestaus sekä UAT eli liiketoiminnan hyväksymistestaus. Itse koodauksen ja yksikkötestauksen tekee toimittaja. Tämän jälkeen moduulit testataan vielä toimittajalla ja toimitetaan meille yhdessä testidokumenttien kanssa. Määritysten mukana menevät toimittajalle testitapaukset ja näiden perusteella tehdään työmääräarvio. H4:n mukaan käytössä oleva testauspolitiikka on SAPin kautta tullut käytäntö. Testausstrategiaa hän ei tunnista, eikä katso sen olevan asiantuntijoiden päätettävissä. H4: ”Meillä on harvoja henkilöitä, jotka antavat luvan tuotantoonvientiin. Käytössä on eräänlainen suppilomalli, jossa moni tekee, mutta vain harva hyväksyy. Mitään ei mene tuotantoon sattumalta”.

3.2.4 Kysymys ”Onko testaustyötä tekevillä käytössään tarvittavat työkalut?”

Tutkimuskysymykseen työkaluista saatiin kaikista kattavimmat vastaukset. Haastateltavat olivat melko yksimielisiä työkalujen puutteista. H1:n mukaan ”nykyisillä regressiotestauksilla ei pystytä todentamaan ollenkaan suorituskykytestausta. Tällä hetkellä testataan vain toiminnallisuutta. Työkalu suorituskykytestaukseen on rakennettu

2013 ja tämän pilottikokeilu oli vuonna 2014, jonka jälkeen jatkosta ei päätetty. Tämän jälkeen sitä ei ole ilmeisesti käytetty. Muita työkaluja ei oikein ole".

H4 kuvailee, että käytössä olevat työkalut ovat olemattomia. Dokumentointi kyllä toimii hyvin täytettynä, mutta prosessi ei pakota täyttämään lomakkeita kattavasti. H2 kertoo työkalujen olevan SAP-testiympäristöt, Microsoft Word ja Excel. Muita työkaluja testauksen suorittamiseen ei hänen mukaan ole. H2: "Erään toimittajan rakentama suorituskykytestausautomaatti on, mutta tämän nykytilasta ei ole tietoa. Tämän käyttö vaatii paljon esityötä". H3 kertoi käyttävänsä soapUI-työkalua XML-rajapintojen testaukseen edellä mainittujen työkalujen lisäksi. Dokumenttien hallintaan ei ole tällä hetkellä mitään työkalua. H3 kertoo "käytössä olevien työkalujen olevan vähäisiä Wordin ja Excelin lisäksi" ja jatkaa, että "suurin osa tehdään manuaalisesti silmällä tarkastaen". H3 kertoo dokumenttien osalta oleva liian monta paikkaa, joihin dokumentit tulee tallentaa, eikä edes muista niitä kaikkia. "Tähän on tulossa loppuvuodesta 2015 uusi muutostenhallinnan ja testaamisen hallintajärjestelmä, jonka pitäisi olla hyvä": kertoo H3.

3.3 Testauksen tahtotila

Haastattelun toisessa osassa kysyttiin, kuinka asioiden tulisi jatkossa kulkea, jotta saataisiin selville miten testausprosessia tulisi parantaa.

3.3.1 Kysymys "Miltä uusi testausstrategia voisi näyttää?"

Vastaukset tutkimuskysymykseen uudesta testausstrategista olivat pitkälti samankaltaisia. Lähes kaikki haasteltavat olivat melko tyytyväisiä nykyiseen malliin ja uskoivat, että siitä saadaan pienillä parannuksilla hyvinkin toimiva.

H1:n mielestä tärkeimpänä tavoitteena on yritys X:n pitkän prosessin osaamisen laajentaminen testaukseen osallistuvan henkilöstön kesken. Tulisi suunnitella, ketkä opettelevat lisää sekä ketkä opettavat ja ketä. Hän toivoisi lisää ymmärrystä päivittäiseen tekemiseen ja jatkokehittämiseen. Edellä mainitut asiat ovat prioriteettilistalla H1:n mielestä korkeimpana. H1 on sitä mieltä, että kaikkien tietojärjestelmien tulisi olla saman testauspolitiikan alla. Samanlainen toimintamalli kaikille järjestelmille ja mikäli se ei

ole mahdollista joltain osin, niin perustelut sille, miksi näin ei voi olla, painottaa H1. Jatkossa hän toivoisi myös kunnollista regressiotestausta. H1 sanoo, ettei ole pidetty kiinni tuotantopakettiajattelusta. Hän ei tunnista, miten muita järjestelmiä testataan, eikä sen vuoksi osaa ottaa kantaa, onko olemassa erilaisia testausstrategioita. Myös H2 piti nykyistä mallia toimivana. Hän nosti kuitenkin esille kysymyksen: ”kuka testausta tekee jatkossa ja kuinka pitkälle se on automatisoitua?” H2:n mukaan olisi hyvä, jos testauksessa päästäisiin 50 % -automaatiotasolle.

H3 toivoisi käytettävän enemmän aikaa suunnitteluun ja erityistä testausta vaativiin uusiin toiminnallisuuksiin. Hallintajärjestelmiä tulisi hänen mukaan olla yksi, jossa kaikki testausta koskevat tiedot säilytettäisiin. Hän pohtii ”pystytäänkö uuden hallintajärjestelmän avulla ohjaamaan testausta niin, että aikataulusta lipsuminen ei enää onnistuisi liiketoiminnan puolelta, vaan myöhässä jätetyt kehitystyöt siirtyisivät automaattisesti seuraavaan mahdolliseen ajankohtaan”.

H6 kertoi suurimmaksi ongelmaksi sen, ettei toimittajalle ehditä suunnitella tarpeeksi hyviä testaustapauksia. Paremman testisuunnittelun seurauksena toimittajalle ei tarvitsisi palauttaa töitä uudelleen työstettäväksi ja testattavaksi. Uudessa testauspolitiikassa tulisi kiinnittää huomio liiketoiminnan osaamisen hyväksikäyttämiseen hyväksymistestauksessa ja kattavien testitapausten luomisessa. Testaukseen liittyvää tekemistä tulisi siirtää liiketoiminnalle ja enemmän myös toimittajalle. IT:n tekeminen tulisi fokusoida enemmän hallintaan, toteaa H6.

Jatkossa H6:n mukaan tarvitaan testausautomaattityökalun keskitetty käyttöönotto. Näin nykyistä työtaakkaa tietohallinnossa kevennettäisiin huomattavasti. Testauksenhallintatyökalun puuttuessa tekeminen hidastuu IT:n toiminnan vuoksi. Testauksen kannalta tärkeimmät asiat tulevat liiketoiminnasta, sanoo H6. Tämä tieto ei kuitenkaan mene aina toimittajalle asti. H6 näkee tärkeimpänä kehityskohteena suunnittelun painoarvon lisäämisen. Hän kertoo, että testauksen tulisi myös säilyä samanlaisena läpi koko prosessin. H6 mainitsee: ”asiakastyytyväisyyskriteerit eli yksikään toimittajalle annettu testitapaus ei palautuessaan sisällä virheitä. Mitä kattavammaksi saadaan toimittajille annettavat testitapaukset, sitä helpommalla itse päästään”.

3.3.2 Kysymys ”Miltä uusi käytännön testausta tukeva prosessi ja aktiviteettiluettelo voisi näyttää?”

H1 toivoisi selkeää prosessia, että ”edellytyksenä UAT:lle on onnistunut moduulitestaus, jonka jälkeen on tehty nämä ja nämä dokumentit sekä ennalta määritetyt tarkastuspisteet hoidettu”. H1, kuten muutkin, haluaa pitää testidokumentit mahdollisimman yksinkertaisena. Testauksen hallintaan H1 toivoisi parempia työkaluja, jotka seuraisivat tehtävien suorittamista. Hänen mielestään testauksessa tarvitaan ohjelmiston suorituskykytyökalu. H1 haluaakin tietää, mitä vuonna 2013 rakennetulle työkalulle tehdään.

H2: ”On alkamassa muutoshallinnan- ja testaamisen kehitysprojekti, josta saadaan apua dokumentointiin ja muutoksen hallintaan. Se ohjaa testaamista. Testisuunnitelma järjestelmän näkökulmasta tulee suoraan täältä. Kun prosessi on kuvattu, ei tehdä turhaa työtä. Keskitytään siis oikeiden asioiden testaukseen”. H2 näkisi testauksen kulun jatkossa järjestelmän ohjaamana, mutta testaajan ei välttämättä tarvitsisi olla itse kehitystyön määrittäjä. Testauksen voisi tehdä myös ulkopuolinen kollega tai joku muu. Näin varmennettaisiin laatua, kun useampi silmäpari olisi mukana testauksessa. Näin toimimalla ei olisi sokeutta omaan tekemiseen. Hän kertoo, että voitaisiin esimerkiksi perustaa erillinen testitiimi. H2 katsoo, että 4-porrasmalli on hyvä myös jatkossa. V-malli toimii tässä meidän testausmaailmassa. Uudet lomakkeet voisivat olla enemmän täyttämistä ohjaavat, nykyiset dokumenttipohjat eivät ohjaa mitenkään, toteaa H2.

H2: ”Tarvitsemme täyttöohjeet, yhtenäinen malli kaikille. Selkeä malli, jotta kuka tahansa voisi tarkastaa miten tämä on mennyt. Varsinkin virhetilanteiden selvittäminen olisi helpompaa. Itse testausmallia ei tarvitse muuttaa. Mallin kehykset ovat toimivat”. Kun muutoshallinnan ja testaamisen hallintajärjestelmä saadaan käyttöön, voitaisiin tutkia testauksen automatisointia, H2 pohtii. Testausautomaatio tulisi ottaa käyttöön ainakin osittain, on paljon perusjuttuja, mitä voi testata. Ainakin Suorituskyky-, regressio- ja integraatiotestaukset voitaisiin hoitaa automaatiolla. Näin asiantuntijoille jäisi enemmän aikaa itse kehittämiseen. Nyt suorittaminen vie paljon aikaa. Laadunvarmistus tulee jatkossa uuden hallintajärjestelmän kautta, kun versionhallinta on kunnossa SAP:ssa. Myös testausmallista tulee selkeämpi, kun hallintajärjestelmä ohjaa prosessia, kertoo H2.

H3 näkee eniten puutteita testisuunnitelmien suunnittelussa. Hänen mukaansa suunnittelun tulisi olla tarkempaa ja sitä pitäisi tehdä enemmän etupainotteisesti. Tuleva uusi hallintajärjestelmä ei käyttöönotettaessa myöskään korvaa kokonaan Word-dokumenttien käyttöä testauksessa. H3 pohtii mahdollisuutta poistaa description- sarake nykyisistä lomakkeista. Hänen mukaansa description- sarake (testitapauksen kuvaus) voitaisiin yhdistää input- sarakkeen (testitapauksen syöte) kanssa. Lisäksi dokumentin näkymän muuttaminen vaakatasoon toisi lisää tilaa kuvakaappauksille. H3 toivoisi uudessa toimintamallissa käytettävän enemmän aikaa itse testauksen suunnitteluun. Hallintajärjestelmiä tulisi hänen mukaan olla vain yksi, jossa kaikki testausta koskevat tiedot säilytettäisiin. Dokumenttipohjat tulisi kaikki yhtenäistää ja muuttaa ohjeistaviksi, jotta kaikki osaisivat täyttää ne samalla tavalla. H3:n mielestä nykyistä dokumentointikäytäntöä tulisi saada sujuvammaksi ja kaikkien tallennusten tulisi sijaita samassa paikassa. Nykyinen käytäntö on byrokraattinen ja vie paljon aikaa. Hänen mukaansa on tärkeää säilyttää nykyinen joustavuus myös tulevassa prosessimallissa.

H3 sanoo, että testauksen perusprosessit tulisi automatisoida ja regressiotestaus pitäisi kokonaan automatisoida. Regressiotestaus voitaisiin myös ajastaa tapahtumaan aina muutosten yhteydessä. Nykyään testausta aikataulutetaan sähköpostilla ja rytmitetään Excelissä. Paljon testauksen liittyvää työtä on muistin varassa. Testausresurssit tulisi myös sitoa kiinni testaukseen sen ajaksi, eikä heitä tulisi käyttää muissa projekteissa samaan aikaan. H3 odottaa uutta testisuunnitelman hallintajärjestelmää ohjaamaan testausta ja dokumenttien hallintaa. Hän toivoo myös testauksen rytmitykseen parempaa mallia, kuten myös aikataulunhallinnan työkalua. Hänen mielestään nykyisessä mallissa on liikaa testauksen liittyviä tehtäviä pelkän muistin varassa.

H4:n mukaan tarvitaan monipuolisemmat testitapaukset siitä, mihin kaikkeen tehtävä muutos voi vaikuttaa. Esimerkiksi eri variaatiot regressiotestauksessa: kuinka muut toimialueet otetaan huomioon, koska prosessit eivät toimi kaikille toimijoille samalla tavalla. H4 sanoo, että testitapaukset voivat toimia paikassa A, mutta eivät esimerkiksi paikassa B. Nämä eri variaatiot tulisi myös testata. Työn tilaajan tulisi aina tehdä hyväksymistestaus. Tietohallinto hyväksyy kehitystyöt joskus siitä syystä, että liiketoiminta ei osaa itse testata niitä. Varsinkin taustatoiminnoissa tapahtuvissa muutoksissa liiketoiminnan kyky testata on huono, kertoo H4.

H6 on samaa mieltä H3:n kanssa siitä, että nykyinen testauslomakepohja sisältää turhan kentän, joka voitaisiin poistaa kokonaan. Hän sanoo, että ”kaikki kehitystyöt tulisi erikseen arvioida regressiotestauksen tarpeelle. Lisäksi liiketoiminnan tuottama testitapauskirjasto tarvitaan kaikille osa-alueille”. H6 haluaisi toimittavan enemmän toimittajalähtöisesti ja määrämuotoisesti, sillä nykyinen toimintatapa vie liikaa aikaa tietohallinnon asiantuntijoilta. Lomakkeet tulee tehdä yksinkertaisiksi ja helpoiksi täyttää. SAPin ulkopuolisten järjestelmien käyttö tulee lisääntymään ainakin CATMAN –osa-alueella, ja siksi testauksenhallintatyökalun tarve korostuu. Kaikelle dokumentaatiolle tulisi olla yksi säilytyspaikka, eikä siellä tulisi olla yhtään ylimääräistä dokumenttia, sanoo H6. Kokonaisuuden hallintaan tulisi ottaa käyttöön yksi kehittynyt järjestelmä. Kehittyneen hallintajärjestelmän välityksellä voidaan jopa kommunikoida toimittajan kanssa. Nykyisestä usean tallennuspaikan dokumentointikäytännöstä on päästävä eroon, painottaa H6.

3.4 Tulosten yhteenveto

Tämän opinnäytetyön tarkoituksena oli tutkia sovellustestausprosessia ja selvittää sen nykyisiä puutteita. Aiheena oli sovellustestaus ja sen hallinta kohdeyrityksen näkökulmasta. Tärkeimpänä tavoitteena oli saada testauksesta määrämuotoista selkeyttämällä testausstrategia testaukseen osallistuville henkilöille. Kokonaistavoitteena oli parantaa yritys X:ssä tehtävää sovellustestausta.

Yhteenvetona haastattelutuloksista voidaan todeta, että parempiin tuloksiin testausprosessissa päästään, kun määrittelyjen yhteydessä luodaan myös toimittajalle annettavat kattavat testitapaukset. Haastatteluissa kävi ilmi, että liiketoimintaa kaivattaisiin mukaan testitapausten suunnitteluun ja luomiseen alusta lähtien. Kun kaikki osapuolet ovat alusta asti tietoisia siitä, mitä ollaan tekemässä, ja kun kaikilla on samanlainen käsitys lopputuloksessa, ollaan oikealla tiellä. Suunnittelussa voidaan käyttää myös järjestelmätoimittajan asiantuntemusta hyväksi. Tärkeintä on kuitenkin suunnitella enemmän etupainotteisesti yhdessä liiketoiminnan ja mahdollisesti myös järjestelmätoimittajan kanssa. Kuten teoriaosuudessa todettiin, ohjelmistoyritykset pitä-

vät tärkeimpänä laadunvarmistuksena huolellista suunnittelu- ja määrittelyvaihetta. Liiketoiminnan ymmärrystä testauksen kokonaisprosessista tulisi lisätä.

Tulisi pohtia voidaanko tarvittavaa osaamista lisätä esimerkiksi testauksen työpajatyöskentelyllä. Pidän osaamisen kehittämistä niin tärkeänä osa-alueena, että otin sen yhdeksi jatkotutkimuskysymykseksi. Liiketoiminnan prosesseista paremmin ymmärtäviä henkilöitä voidaan käyttää itse testauksessa tai kattavampien testitapausten suunnittelussa ja tietohallinnon konsultoinnissa. Kuten H2 kertoi omassa ryhmässään jo tapahtuvan testauksen osalta. Mielestäni tätä mallia tulisi kopioida, ja se tulisi ottaa myös muiden ryhmien toimintatavaksi. Tätä mallia voidaan vielä laajentaa ottamalla järjestelmätoimittajan edustaja mukaan yhteisiin suunnittelutapaamisiin.

Tuloksista kävi ilmi, kuinka tärkeää on liiketoiminnan ja tietohallinnon sitoutuminen yhteiseen päämäärään, joka tässä tapauksessa on tarkoituksenmukainen järjestelmätoiminto tai sovellus. Tuloksia täydennettiin vielä haastattelemalla järjestelmäpäällikkö H7. Hänen vastauksiaan käytettiin lähinnä testauspolitiikan ja – strategian luomisessa. Muutaman hänen huomionsa voisi tässä kuitenkin mainita. H7:n mukaan tulee luoda eräänlainen peruutussuunnitelma: ”Mitä tehdään silloin, kun jo tuotantoon viedyssä työssä huomataan virhe”. Hän painotti myös laatumittareiden saamisen tärkeyttä: ”mitä ei voida mitata, sitä ei voida johtaa”. Hän toivoi myös ekstra-silmäparia tekemiseen, ja ratkaisuksi tähän hän haluaisi tietohallinnon operatiivisesta toiminnasta vastaavan ryhmän osallistuvan testaukseen.

Muut huomion arvoiset ja testauspolitiikkaan kirjattavat asiat:

- Yksi tallennuspaikka kaikille testaukseen liittyville dokumenteille.
- Yksi universaali kieli testaukseen ja testausdokumentteihin, joka on englanti.
- Kaikkien järjestelmien tekeminen saman testauspolitiikan alle soveltuvin osin.
- Peruutussuunnitelma: kuinka jo tuotantoon viedyt virheelliset työt otetaan pois sieltä.
- Testauksessa käytettävien lomakkeiden pohjat täyttöohjeineen.

Asiantuntijat olivat harvinaisen yksimielisiä halutessaan yhtenäisen suunnitelman, projektista tai järjestelmästä riippumatta. Kaikkien järjestelmien tekeminen on saatava

yhtenäiseksi. Kun eri järjestelmien prosessit, työkalut, lomakkeet ja dokumentointikäytäntö on samannäköistä, myös testitulosten vertailu ja ymmärtäminen helpottuu. Haastatteluun osallistuneet odottivat myös paljon tulossa olevalta uudelta testauksen hallintasovellukselta.

Työn lopputuloksena syntyi lisäksi uuden sovellustestausprosessin kulkua kuvaava prosessikuvaus (liite 2.), esimerkkimallit täyttöä ohjaavista testauslomakkeista. Vaikka testauksenhallintaan onkin tulossa työkalu, se ei täysin poista Word- ja Excel-pohjien käyttöä testauksessa. Lisäksi luotiin otsikkotasoiset, noin kolmeenkymmeneen kysymykseen vastauksen antavat, testauspolitiikka ja –strategia IT:lle jatkotyöstettäväksi. Työn ensisijaisena tavoitteena oli tehdä yritys X:n sovelluskehitysprosessista määrämuotoisempaa ja selkeyttää testausstrategia testaukseen osallistuville. Tutkimuksen pohjalta luodut dokumentit ovat testausstrategia sekä –politiikka. Testauspolitiikassa ja –strategiassa otetaan kantaa moneen aikaisemmin epäselvään kysymykseen, joten uskon tavoitteen tältä osin täyttyvän. Luottamuksellisista syistä näitä ei kuitenkaan julkaista työn tuloksissa. Työn ajankohta oli osuva, sillä työn pääasiallisia tuloksia eli testauspolitiikkaa ja –strategiaa päästään kokeilemaan heti uuden testauksenhallintajärjestelmän kehitysprojektissa.

4 Pohdinta

Tutkin opinnäytetyössä yritys X:n sovellustestausprosessia. Opinnäytetyön tavoitteena oli tehdä kohdeyrityksen testauksesta määrämuotoista ja laadukkaampaa selkeyttämällä testausstrategia sekä testauskäytännöt koko organisaatiolle. Henkilökohtaisena tavoitteenani oli tutkimustyön ohella saada parempi käsitys ohjelmistotestauksesta ja sen hallinnasta. Opinnäytetyössä selvitettiin, kuinka sovellustestauksen kokonaisprosessi toimii tällä hetkellä testaustyötä tekevien mielestä. Haastattelujen perusteella nykyisen dokumentointikäytännön katsottiin olevan suurin ongelma tällä hetkellä. Tämä puute olikin jo huomattu, ja sitä oli tulossa korjaamaan uusi testauksenhallintajärjestelmä. Kun uusi järjestelmä saadaan käyttöön, ja dokumentointiin luodaan yhteiset ohjeet, tämä osa-alue tulee olemaan toivotulla tasolla ainakin haastattelujen perusteella.

Mielestäni tutkimuskysymyksiin saatiin riittävät vastaukset haastatelluilta. Vastaukset ovat myös teoriaan verrattuna uskottavia. Pidän tuloksia luotettavana tutkimuksen näkökulmasta katsottuna, sillä kuusi pitkään testausta tehnyttä haastateltavaa on mielestäni riittävän kattava otos toimeksiantajan ja työn tarpeisiin. Mielestäni yllättäviä tuloksista tekivät haastateltujen yksimielisyys ja kriittisyys testauksen nykytilaan. Moniin haastattelukysymyksiin saatiin lähes identtinen vastaus henkilöstä riippumatta. Haastateltavat olivat melko kriittisiä nykyisiin käytäntöihin ja työkaluihin, vaikka itse testausprosessi heidän mielestään sujuikin melko hyvin. Mielestäni haastatellut kokivat, että asiat voitaisiin tehdä paljon paremmin pienillä muutoksilla. Haastatteluissa esille tulleet ongelmat voivat vaikuttaa ulkopuoliselle hankalasti korjattavilta, mutta en itse koe asian olevan niin. Useat tutkimuksessa ilmi tulleet ongelmakohdat ovat yhteisesti sovittavissa ja korjattavissa pienillä muutoksilla toimintatavoissa. Uusiin toimintatapoihin, kuten dokumentointikäytäntöön tullaan tekemään erillinen ohjeistus.

Pidän tuloksia uskottavia ja hyvin yleistettävinä. Samanlaisiin ongelmiin olen tutustunut opintojeni varrella moneen otteeseen. Mielestäni tulokset heijastavat tietoperustassa esitettyjä asioita kattavasti. Myös testauksen psykologinen näkökulma tulee esille tuloksissa. Ihmiset ovat hyvin päämäärätietoisia ja tarvitsevat tavoitteita. Lisäksi parhaiten virheitä ohjelmistosta löytää ohjelmistokehitykseen osallistumaton henkilö. Toimivien dokumentointikäytäntöjen tarpeellisuutta korostetaan teoriaosuudessa moneen ottee-

seen. On siis täysin ymmärrettävää, että erilaiset dokumentointikäytännöt ja oleellisten testausdokumenttien puuttuminen tekevät testauksesta tarpeettoman työlästä. Työn tuloksiin on kuitenkin suhtauduttava kriittisesti, kun tarkastellaan yritys X:n sovellustestauksen kokonaiskuvaa, sillä mukaan ei otettu kaikkia prosessiin osallistuvia, vaan työn rajauksen ja osittain aikataulun vuoksi haastateltiin vain IT:ssä työskenteleviä.

Opinnäytetyötä oli alkuhaasteiden jälkeen varsin mieluisaa tehdä. En tiennyt ohjelmistotestauksesta työn suunnitteluvaiheessa kovinkaan paljoa, joten oikean kirjallisuuden löytäminen juuri tähän aiheeseen ei ollut helppoa. Tämän tutkimustyön tekemisestä ja testauksen perusluonteen ymmärtämisestä on varmasti paljon hyötyä tulevaisuudessa: toimin sitten missä tahansa organisaatiossa tai yrittäjänä. Ohjelmistoja käyttäviä tietokoneita ja älylaitteita kun tarvitaan kaikkialla. Alussa ongelmia oli eniten työn rajauksen kanssa, ja työ olikin vaarassa paisua monta kertaa liian suureksi ja hallitsemattomaksi kokonaisuudeksi. Tutkimuksen aiheen valinta tapahtui jo loppuvuodesta 2014, kun minulle tarjottiin tätä aihetta. Erilaisten muodollisuuksien vuoksi itse kirjoitustyö päästiin aloittamaan vasta tammikuun 2015 lopussa. Työn teoriaosuuden kirjoittaminen oli erittäin työläs vaihe ja siihen kului paljon suunniteltua enemmän aikaa. Myös haastatteluiden suunnittelu, toteuttaminen ja tulosten purku nauhoituksista vei useita viikkoja.

Opinnäytetyön tekeminen opetti minulle paljon myös muusta testaukseen välillisesti liittyvästä tekemisestä ja projektinhallinnasta. Olen tyytyväinen aiheen valintaan, koska opin paljon ohjelmistotestauksesta ja tutkimuksen suunnittelusta sekä toteutuksesta. Olisin kuitenkin toivonut voivani olla enemmän mukana tietohallinnossa työn aikana. Lisäksi haastatteluun olisi tullut valita myös järjestelmätoimittajan edustaja sekä haastatella, miten testaus näkyy liiketoiminnalle. Olen myös paremmin valmistautunut tavoitteisiini tulevissa tietohallinnon työtehtävissä. Mikäli työ aloitettaisiin uudestaan, tekisin varmasti asiat eri tavalla ja valitsisin vain yhden ongelman ratkottavaksi kerrallaan. Kaikkein haastavinta tässä opinnäytetyössä oli kuitenkin työn tekemisen yhdistäminen muihin opintoihin ja asiantuntijan päivätöihin. Monesti tuntui siltä, ettei vuorokaudessa vain ole tarpeeksi tunteja kaikkien asioiden tekemiseksi. Työ saatiin kuitenkin valmiiksi aikataulun mukaisesti. Opinnäytetyön kirjoittamisen jälkeen päästiin todellisten ongelmien pariin, eli hyödyntämään tuloksia käytännössä.

4.1 Jatkotutkimuskohteet

Opinnäytetyön tekemisen aikana kohtasin useita mahdollisia jatkotutkimuskohteita. Vahvimmin esille kuitenkin nousivat neljä seuraavaksi esiteltyä aihetta. Kohteet tulivat toistuvasti esille myös haastatteluissa. Kaikissa haastatteluissa mainittiin jollain tapaa testausautomaation, testaustyökalujen sekä dokumenttienhallintaan liittyvät puutteet. Testauksen- ja dokumenttienhallintaan on tulossa apua uuden järjestelmäprojektin muodossa, joten siihen ei tarvinne kiinnittää huomiota jatkotutkimuksissa.

Ensimmäisenä jatkotutkimuskohteena nousee vahvasti esille testausautomaation mahdollisuuksien selvittäminen. Mielenkiintoisena tämän aiheen kokivat myös haastatellut, sillä lähes kaikki haastellut toivoivat varsinkin regressiotestauksen automatisoinnin lisäämistä. Automaation tarve vain korostuu tulevaisuudessa, kun testattavien prosessien määrä lisääntyy. Varsinkin regressiotestauksessa automaatiolle olisi kysyntää. Haastattelussa H5 kuitenkin totesi, että manuaalisesti suoritettava testaus on ensin saatava riittävälle tasolle, joka toivotaan saavutettavan uuden testauksenhallintajärjestelmän avulla ja dokumentointikäytäntöjen muutoksilla. Vasta sitten voidaan harkita automaation käyttöönottoa testauksen tukena.

Toinen suosittelemani jatkotutkimuskohde on testaustyökalut ja niiden käyttö. Haastatelluista vain H3 kertoi käyttävänsä jotain erillistä sovellusta testauksen apuna. Hänellä oli käytössä avoimen lähdekoodin soapUI -työkalu. Tulisi selvittää, voisivatko myös muut käyttää vastaavanlaisia testaustyökaluja apunaan. Muutamat haastatellut halusivat selvittää olemassa olevan suorituskykytyökalun nykyisen statuksen ja mahdollisen käyttöönoton. Kolme haastatelluista mainitsi tämän työkalun, mutta ei tiennyt tarkalleen, käytetäänkö sitä vakituisesti tai missä tilanteissa sitä olisi tarkoitus käyttää.

Kolmas mainitsemisen arvoinen jatkotutkimuskohde on testausosaamisen jakaminen erityisesti tietohallinnossa, mutta myös testaukseen osallistuvien liiketoiminnan edustajien kesken. Myös haastattelutuloksista ilmeni tarve testauskoulutussuunnitelmalle. Osaamisen henkilöityminen katsottiin ongelmaksi varsinkin pitkään testausta tehneiden osalta. Tulisi kehittää suunnitelma, jolla uusia testaajia koulutettaisiin ja vanhojen osaa-

jien tietotaitoa pystyttäisiin jakamaan organisaatiossa. Näin toimimalla varmistettaisiin organisaation osaaminen nyt ja tulevaisuudessa.

Neljäs, ja omasta mielestäni tärkein tutkimuskohde, olisi testauksen ja sovelluskehityksen mittarit. Selkeiden testaustyötä ja sen laatua mittaavien mittarien puuttuminen kävi ilmi myös haastatteluissa. Mittareita tarvitaan myös toiminnan kehittymisen arviointiin.

Lähteet

Blair M, Obenski S, Bridickas P. 1992. Patriot missile system failure report. Luettavissa: http://www.fas.org/spp/starwars/gao/im92026.htm. Luettu 21.3.2015

Helfen Markus, Trauthwein Hans Martin 2011. Testing SAP solutions. 2nd edition. Bonn, Boston. Galileo Press.

Hoffman David 1999. I Had A Funny Feeling in My Gut. Luettavissa: http://www.washingtonpost.com/wp-srv/inatl/longterm/coldwar/shatter021099b.htm. Luettu 22.3.2015

International Atomic Energy Agency 2001. Investigation of an accidential exposure of radiotherapy patients in Panama. Luettavissa: http://www-pub.iaea.org/MTCD/publications/PDF/Pub1114_scr.pdf. Luettu: 21.3.2015

Itkonen Juha 2011. Empirical studies on exploratory software testing. Luettavissa: http://lib.tkk.fi/Diss/2011/isbn9789526043395/isbn9789526043395.pdf. Luettu: 3.4.2015

Kasurinen Jussi Pekka 2013. Ohjelmistotestauksen käsikirja. 1 painos. Docendo. Jyväskylä.

Kit Edward 1995. Software Testing in the Real World: Improving the process. Addison – Wesley. Reading, MA.

Levenson Nancy, Turner Clark S. 1993. An Investigation of the Therac-25 Accidents. Luettavissa: http://courses.cs.vt.edu/professionalism/Therac_25/Therac_1.html. Luettu: 21.3.2015

Lions, J. L. 1996. ARIANE 5 Flight 501 Failure: Report by the Inquiry Board. Luettavissa: http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html. Luettu 21.3.2015

Myers, Glenford J., Corey Sandler, Tom Badgett 2011. The Art Of Software Testing. 3rd edition. John Wiley & Sons, Inc., Hoboken, New Jersey.

Office of Government Commerce 2007. ITIL – Service Transition. TSO. Norwich, Iso-Britannia.

Perttu Jukka 2013. VTT uskoo Suomen mahdollisuuksiin teollisessa internetissä. Helsingin Sanomat. Luettavissa: http://www.hs.fi/talous/a1385966852021. Luettu: 8.3.2015

Tassey Gregory 2002. The Economic Impacts of Inadequate Infrastructure for Software Testing. Luettavissa: http://www.nist.gov/director/planning/upload/report02-3.pdf. Luettu: 2.3.2015

U.S General Accounting Office 1981. NORAD's Missile Warning System: What Went Wrong?. Luettavissa: http://archive.gao.gov/f0102/115265.pdf. Luettu: 21.3.2015

Liitteet

Liite 1. Teemahaastattelun kysymysrunko

Kysymykset sovellustestauksen nykytilan selvittämiseksi

Miten nykyiset testisuunnitelmat toimivat?

Miten nykyiset testitapaukset tyypillisesti toimivat?

Mitä koet nykyisessä testausmallissa toimivaksi?

Minkälaisia puutteita tunnistat nykyisessä testausmallissa?

Kuinka nykyiset testilomakkeet mielestäsi toimivat?

Mitä ajattelet nykyisistä testaustyökaluista?

Minkälainen on nykyinen dokumentointikäytäntö?

Minkälainen on yritys X:n nykyinen testauspolitiikka?

Minkälainen on tämän hetkinen testausstrategia?

Kysymykset kehityskohteiksi ja tahtotilan selvittämiseksi

Miten testisuunnitelmaa tulisi muuttaa?

Miten testitapauksia tulisi muuttaa?

Miten testaus tulisi jatkossa tehdä?

Mitä nykyisestä testausmallista tulisi säilyttää myös uudessa mallissa?

Minkälaisena näkisit uudet testilomakkeet?

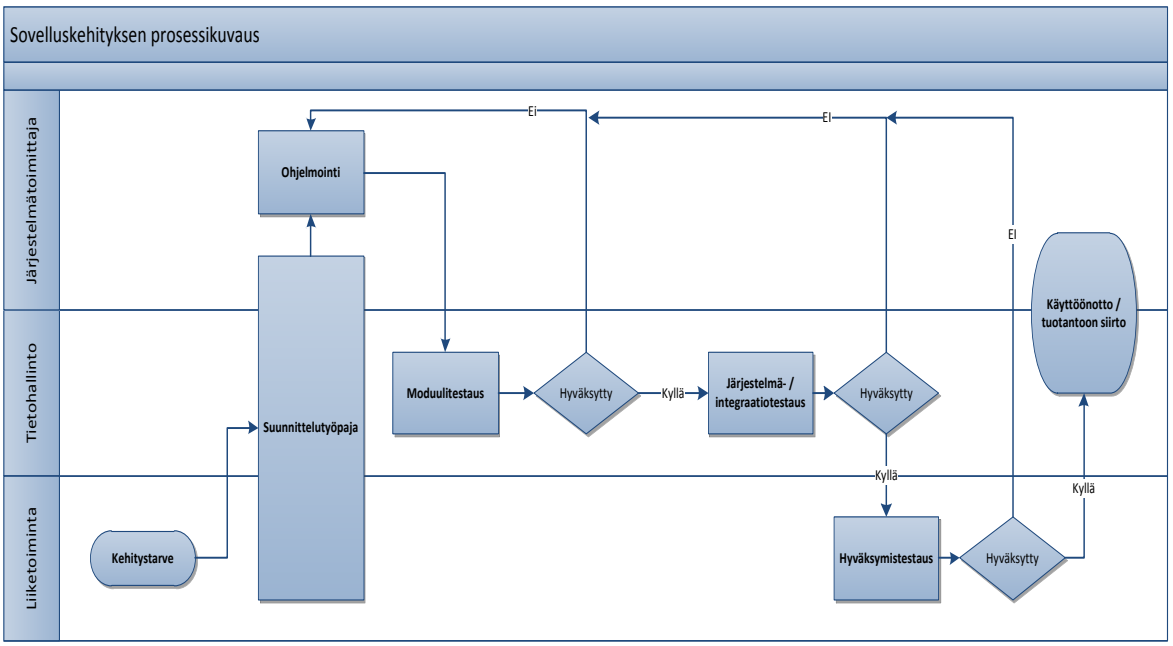
Minkälaisia testauksen työkaluja näkisit jatkossa tarvittavan?

Miten dokumentointikäytännön tulisi jatkossa kulkea?

Minkälaisena näkisit uuden testauspolitiikan?

Minkälaisena näkisit uuden testausstrategian?

Liite 2. Sovelluskehityksen prosessikuvaus



Liite 3. Käytetyt termit ja lyhenteet

ASAP implementation roadmaps, SAPin käyttöönotoissa käytettävä prosessimalli.

CATMAN eli Category Management, valikoimien hallinta.

EhP (Enhancement Package) on SAPin parannuspaketti eli version vaihto.

Implementointi sanaa käytetään kun ohjelmistoja otetaan käyttöön.

Intranet on organisaation sisäiseen käyttöön tarkoitettu lähiverkko.

ISO/IEC 29119 on kansainvälinen ohjelmistotestauksen standardi.

ISTQB (International Software Testing Qualifications Board) myöntää serfikaatteja ja kouluttaa testaajia.

Järjestelmä nimitystä käytetään kun kyse on ohjelmiston ja laitteiston muodostamasta kokonaisuudesta.

Komponentti on kokonaisuuden osa, itsenäinen ja uudelleenkäytettävä ohjelmistoyksikkö.

Ohjelma on tietokoneohjelma ja siihen liittyvä dokumentaatio.

Ohjelmointirajapinta on määritelmä, jonka mukaan ohjelmat voivat vaihtaa tietoja keskenään.

SAP ® AG on Saksalainen ohjelmistoyritys, joka on erikoistunut ERP- eli toiminnanohjausjärjestelmiin.

soapUI on avoimen lähdekoodin työkalu ohjelmointirajapintojen (API) testaukseen.

Sovellus eli sovellusohjelma on tietokoneohjelma, joka on suunniteltu tiettyä käyttötarkoitusta varten.

Testitapaus tarkoittaa syötearvojen, esiehtojen, odotettujen tulosten ja suorituksen jälkiehtojen muodostamaa kokonaisuutta, joka on muodostettu tiettyä tavoitetta tai testauksen kohdetta varten.

XML (Extensible Markup Language) on merkintäkielien standardi, jolla tiedon merkitys voidaan kuvata tiedon sekaan.